Université de **PARIS 7 - Denis Diderot**

Vers une Théorie du Chiffrement Symétrique

Thèse d'Habilitation *7 Janvier 1999*

Président du Jury :
Gilles Kahn

Rapporteurs :
Daniel Krob
Jean-Jacques Quisquater
Claus Peter Schnorr

Examinateur :
Jean Vuillemin

Directeur de Recherches :
Jacques Stern

Serge Vaudenay
Laboratoire d'Informatique de
l'Ecole Normale Supérieure

"The problems of cryptography and secrecy systems furnish an interesting application of communication theory. In this paper a theory of secrecy systems is developed. The approach is on a theoretical level and is intended to complement the treatment found in standard works on cryptography. There, a detailed study is made of the many standard types of codes and ciphers, and of the ways of breaking them. We will be more concerned with the general mathematical structure and properties of the secrecy systems."
Claude E. Shannon
In *Communication Theory of Secrecy Systems*, 1949.

Avant Propos

Ce document constitue le dossier en vue d'une d'habilitation à diriger des recherches soumis à l'Université de Paris 7. Il est constitué

1. d'un chapitre d'introduction (pp. 5–13),
2. d'une première partie qui effectue un travail de synthèse dans le domaine du chiffrement symétrique (en français), illustrée de quelques uns de mes résultats (pp. 15–100),
3. d'un ensemble d'annexes qui sont la copie *in extenso* de mes articles les plus significatifs (en anglais), dont certains sont mentionnés dans la première partie (pp. 101–251). Leur mise en page a pu être légèrement modifiée dans un but purement éditorial et complétée par une courte note préliminaire, parfois par un *erratum*. Leurs bibliographies ne paraissent pas car elles ont été fusionnées avec celle du mémoire (pp. 253–262),
4. d'une liste de notations et d'un index pour faciliter l'accès au document (pp. 263–275),
5. d'un *curriculum vitæ* qui contient notamment une liste de publications complète (pp. 275–280).

On pourra se référer à la table des matières située à la fin de ce document.

Mes recherches concernent la cryptologie : aussi bien la cryptographie que la cryptanalyse. Le domaine de la clef secrète (chiffrement symétrique), représente une part plus importante de mon travail et sera donc le thème essentiel de mon mémoire. J'ai également contribué au thème de la clef publique (chiffrement asymétrique) pour lequel on trouvera en annexes trois articles. Je ne consacrerai à ce domaine qu'une brève note consignée dans le chapitre d'introduction.

Introduction

La cryptologie, définie par Jacques Stern comme "science du secret" est vieille de plusieurs millénaires d'histoire. Ses fondements modernes ne datent cependant que de la révolution technologique, avec l'arrivée du télégraphe, de l'ordinateur, *etc.* Ce sont en fait la seconde guerre mondiale, puis la guerre froide qui ont le plus contribué à son développement, mais sous la forme d'une "cryptologie d'Etat". Dans les années 1970, les américains et les soviétiques rivalisaient dans une course technologique pour la conquête de l'espace et de l'armement. De manière symptomatique, le journal télévisé commençait en France par un gros plan sur un mystérieux téléscripteur présenté sous cloche. L'information, tout comme la technologie, était alors le privilège de l'Etat, dans les pays industrialisés. Dans les années 1970, la cryptologie devint également un réel besoin industriel, et l'Etat américain proposa un procédé de chiffrement standard — DES[1], fondé sur la notion de cryptographie à clef secrète, où le chiffrement et le déchiffrement utilisent une même clef (ce qui les rend symétriques).

La recherche publique en cryptologie prit à cette époque de l'ampleur sous l'impulsion de Whit Diffie, de Martin Hellman et de quelques autres pionniers qui inventèrent la notion de "clef publique". Dans ce nouveau procédé, on utilise des clefs différentes pour le chiffrement (une clef publique) et le déchiffrement (une clef secrète), ce qui les rend asymétriques. La recherche s'orienta au début vers des théories "conceptuelles" comme la théorie des nombres, celle de la complexité algorithmique et l'algèbre moderne, et prit une telle importance que l'on créa en 1981 la première édition de la série de colloques Crypto.

De leur coté, les chercheurs en chiffrement symétrique se focalisèrent dans les années 1980 sur les mystérieuses propriétés combinatoires des boîtes S de DES sans que ces recherches ne débouchent sur de véritables applications. Cela permit toutefois de développer d'élégantes théories mathématiques, mais rendit ce secteur encore plus déroutant.

Lorsque j'ai commencé mon doctorat en 1991, je me suis intéressé à ce

[1] *Data Encryption Standard.*

domaine hermétique qui connaissait de profonds changements. Je fus étonné de constater que l'on pouvait facilement se lancer dans l'attaque de fonctions cryptographiques sans l'aide de théorie véritablement profonde, mais avec tous les moyens du bord (rigoureux ou non). De telles attaques avec des manipulations "au niveau du bit" ressemblaient plus à des solutions de casse-tête mathématiques. D'ailleurs, Kevin McCurley, actuel président de l'IACR[2] résuma de manière provocante l'aspect anarchique de ce secteur en parlant de "*dirty bit tricks*".

A la même époque, Eli Biham et Adi Shamir s'acharnaient à attaquer DES et développèrent la technique de cryptanalyse différentielle. D'autres attaques de caractère général suivirent quelques années après, et il y eut une véritable volonté de poursuivre ces recherches. Ross Anderson créa ainsi en 1993 une série de colloques annuels dédiée à ce domaine : *Fast Software Encryption*.

Vingt cinq ans après l'apparition de DES, l'information n'est plus le privilège des Etats, mais est accessible à tous par Internet, le téléphone cellulaire, la télévision numérique, ... L'économie connaît en ce moment une forte reprise sous l'influence, notamment, des secteurs qui créent des produits immatériels comme le service ou le logiciel. Les produits cryptologiques, enfin, sont des biens déployés massivement, que l'on retrouve tous dans nos portefeuilles (sous la forme de cartes à puce), nos téléphones ou nos disques durs. Symbole d'une époque révolue, DES est maintenant abandonné par le gouvernement américain, et la Chambre de Commerce a lancé un nouveau processus de standardisation, ouvert et international cette fois, qui aboutira à un procédé AES[3].

Le moment est donc idéal pour tirer un bilan de vingt cinq ans de recherches sur le chiffrement symétrique. Bien que de nombreux chercheurs considèrent ce secteur comme s'il était resté à un stade empirique, je pense au contraire qu'il est à présent possible d'ébaucher une "théorie du chiffrement symétrique". L'ambition de ce mémoire est d'en faire l'essai tout en présentant mes propres travaux.

Outre la séparation entre clef secrète (chiffrement symétrique) et clef publique (asymétrique), on fait *stricto sensu* une distinction entre la *crypto-*

[2] *International Association for Cryptologic Research*. La recherche en cryptologie est gérée par l'IACR. Cette association organise les conférences Crypto (depuis 1981) et Eurocrypt (depuis 1982) — à présent annuelles —, et édite le *Journal of Cryptology* qui est la référence du domaine. D'autres conférences importantes sont organisées en parallèle, comme Asiacrypt (qui sera d'ailleurs bientôt gérée par l'IACR) et le colloque annuel *Fast Software Encryption*.

[3] *Advanced Encryption Standard*.

graphie, dans laquelle on propose des algorithmes pour protéger l'information, et la *cryptanalyse*, où l'on cherche à démontrer que l'on obtient une réelle sécurité ou non. Les concepteurs des algorithmes cherchent cependant à démontrer aussi leur sécurité en même temps qu'ils les proposent. L'usage réserve donc plutôt le terme de *cryptanalyste* aux "casseurs d'algorithmes" et considère la preuve de sécurité comme faisant partie de la cryptographie.

Ce mémoire présente ma contribution au domaine de la clef secrète, aussi bien en cryptographie qu'en cryptanalyse. J'ai également contribué au domaine de la clef publique, principalement en tant que "casseur", et trois de mes résultats sont présentés en annexes. Comme ce thème ne sera plus abordé dans la suite, il convient de faire ici un bref commentaire.

Le problème de la cryptographie à clef publique est que les algorithmes doivent reposer sur un procédé "à sens unique" : une fonction mathématique simple doit permettre de calculer une clef publique au moyen d'une clef secrète sans que l'opération inverse soit faisable en pratique. En outre, ces fonctions doivent être "maniables", afin de permettre le chiffrement et le déchiffrement d'un message. De nombreux exemples de telles fonctions sont inspirés de la théorie de la complexité et de la théorie des nombres.

Au début de l'histoire de la cryptographie à clef publique, les chercheurs se sont notamment intéressés au problème dit du "sac à dos". Dans ce procédé, une clef publique est une liste de nombres $k_1, \ldots, k_n$. Un message est représenté sous la forme d'un sous-ensemble M de $\{1, \ldots, n\}$, et l'on chiffre en effectuant la somme des k_i pour $i \in M$. Comme ce calcul est très simple, l'algorithme est performant. Le problème est de dissimuler un procédé pour déchiffrer par le biais d'une clef secrète. On a ainsi une fonction à sens unique qui à cette clef associe la liste $k_1, \ldots, k_n$. Le premier système de Ralph Merkle et Martin Hellman a été cassé par Adi Shamir. Les autres variantes fondées sur un problème semblable ont toutes été cassées, sauf l'un des premiers systèmes présenté à la conférence Crypto 84 et dû à Benny Chor et à Ron Rivest. Bien que la sécurité de l'algorithme de Chor-Rivest ait laissé des doutes, celui-ci est resté "en vie" jusqu'en 1998 au moment où j'ai présenté une attaque contre lui.

La notion de permutation birationnelle constitue une autre idée, proposée par Adi Shamir à Crypto 93, qui aurait également permis d'obtenir un système performant. Ici, la fonction à sens unique est une fonction rationnelle de plusieurs variables. L'idée principale est que cette fonction est choisie pour être réversible, mais la fonction inverse est cachée par des transformations linéaires. Don Coppersmith, Jacques Stern et moi-même avons aussitôt cassé ces algorithmes en développant de nouvelles méthodes d'analyse qui utilisent

des notions de théorie de Galois et des techniques de résolution d'idéaux de polynômes.

Les sacs à dos et les permutations birationnelles étaient proposés comme alternatives aux problèmes de la factorisation (ici, la fonction à sens unique est simplement la fonction de multiplication), et du logarithme discret (où la fonction à sens unique associe à x une valeur $g^x \bmod p$) qui sont utilisés dans presque tous les procédés actuels. DSS[4] est un standard américain de signature numérique inspiré du procédé de signature de ElGamal qui utilise le problème du logarithme discret. L'algorithme dépend de la fonction de hachage SHA[5]. En fait, le message x à signer n'intervient dans le calcul que par la valeur $\mathrm{SHA}(x)$. Il est donc important pour la sécurité qu'il soit difficile de trouver une "collision" $\mathrm{SHA}(x) = \mathrm{SHA}(y)$ avec $x \neq y$. (Dans le cas contraire, on pourrait utiliser une signature d'un message x comme une signature valide d'un message y.) J'ai remarqué que les collisions pouvaient également intervenir au cours de l'utilisation de SHA et non sur la fonction SHA même. En effet, DSS n'utilise que la valeur $\mathrm{SHA}(x) \bmod q$, et l'on peut très bien avoir des collisions sur cette valeur. Si ce résultat paraît anecdotique, il ébranle cependant la solidité de DSS au moment où il est en phase de normalisation par l'ISO[6].

En fait, ces trois résultats ne détruisent pas l'effort des auteurs des algorithmes cassés, mais apportent une meilleure compréhension des problèmes mathématiques sous-jacents. Il est intéressant de noter que ce fonctionnement un peu ludique de la cryptographie débouche en fin de compte sur des idées profondes qui permettent d'isoler ce qui fait réellement la sécurité d'un algorithme. Le secteur de la clef publique a été récemment bouleversé par des chercheurs (notamment mes collègues David Pointcheval et Jacques Stern) qui ont développé des techniques de preuve pour ces algorithmes. D'ailleurs, je me suis servi de ces résultats, dans un travail en collaboration avec David Pointcheval, pour démontrer la sécurité d'une légère variante de DSS (qui élimine donc en particulier mon attaque mentionnée plus haut). Ironie du sort, cette variante de DSS est plus proche de l'algorithme de signature de Claus Schnorr dont le brevet avait été détourné pour définir DSS ! (Cette variante de DSS a été ajoutée au projet final de norme ISO 14888 sous le nom d'"algorithme de Pointcheval-Vaudenay".)

Ce tournant de la recherche dans le domaine de la clef publique est le résultat d'une orientation délibérée. Dans son ouvrage *"La Science du Se-*

[4] *Digital Signature Standard.*
[5] *Secure Hash Algorithm.*
[6] *International Organization for Standardization.*

cret", Jacques Stern distingue les trois âges artisanal, technique et scientifique de l'histoire de la cryptologie (dans ce dernier cas, il parle d'ailleurs d'"âge paradoxal"). Autrefois privilège de quelques "artisans", la cryptologie est devenue l'arme technologique d'une guerre militaire, stratégique puis industrielle où différents procédés, attaques et contre-mesures se sont succédées. La cryptologie évolue ensuite plus scientifiquement en consolidant ses propres bases, ce qui lui permet enfin d'appréhender la sécurité dans un cadre formel rigoureux.

Que cette évolution soit naturelle ou non, j'ai hérité, en tant qu'ancien étudiant de Jacques Stern, de cette démarche systématique que j'ai cherché à appliquer au chiffrement symétrique.

Le premier chapitre présente donc la problématique du chiffrement symétrique ainsi que les méthodes classiques de construction. On observe que la plupart des algorithmes reposent sur la notion de "réseaux de substitutions-permutations" : le procédé de chiffrement suit le circuit d'un réseau au travers de "boîtes de calcul". On distingue notamment la construction suivant le schéma de Feistel qui permet de construire un procédé réversible (ce qui est indispensable pour permettre un déchiffrement) au moyen de fonctions non nécessairement réversibles. D'autres types de construction, comme la fonction Safer de Jim Massey, utilisent des couches de transformations réversibles effectuées en parallèle. Dans ce chapitre, on définit donc la notion formelle de "réseau de calcul" qui sera le fil conducteur des résultats présentés dans ce mémoire.

Le second chapitre traite des méthodes d'attaques qui ont véritablement donné son impulsion à ce domaine de recherche. On ne présente pas les attaques dédiées, *artefacts* d'âges anciens, et l'on se contente de décrire les méthodes générales que l'on peut exploiter dans un grand nombre de contextes. On définit la cryptanalyse différentielle de Eli Biham et Adi Shamir et la cryptanalyse linéaire de Mitsuru Matsui. Ces méthodes d'attaques sont illustrées respectivement par deux de mes travaux : une attaque différentielle de la fonction Blowfish "par clefs faibles" et une attaque linéaire "manquée" de Safer. (L'attaque est manquée en ce sens qu'elle est évitée de justesse grâce à une propriété inattendue. Elle permet cependant de dévoiler une faiblesse potentielle de Safer, et montre une erreur à ne pas commettre dans le développement d'autres algorithmes.) Dans ces exemples, ma contribution ne s'est pas limitée à adapter des idées générales à ces algorithmes, mais a également apporté, dans le premier cas, la notion d'"attaque par clefs faibles", et dans le second une analyse mathématique plus rigoureuse de la cryptanalyse linéaire et l'idée de "multipermutation" développée dans

un autre chapitre. On présente également les méthodes génériques de recherche exhaustive et d'"attaque dans le milieu" qui se généralisent dans un cadre formel défini par Claus Schnorr et moi-même. On généralise également toutes ces méthodes dans des attaques de type statistique qui m'ont permis d'améliorer les meilleures attaques connues contre DES.

Dans un troisième chapitre, on montre les différentes méthodes d'approche pour construire des algorithmes de chiffrement sûrs. Tout d'abord, on présente les méthodes heuristiques (par contrôle de la diffusion de l'information des calculs, et par contrôle de la non-linéarité des fonctions) qui proviennent directement des leçons enseignées par les attaques du second chapitre. On définit notamment la notion de multipermutation qui assure une bonne diffusion au sein des réseaux de calcul. On expose ensuite des méthodes qui permettent de prouver une sécurité face à une classe d'attaques. On présente ainsi les résultats de la "théorie de la décorrélation" dont je suis l'auteur. Ces résultats établissent un lien logique entre des résultats qui semblent *a priori* éloignés : la notion de sécurité de Shannon qui provient de la théorie de l'information, celle de Mike Luby et Charles Rackoff importée du domaine des algorithmes pseudo-aléatoires, les notions de hachage universel de Carter et Wegman qui proviennent de la théorie de la "dérandomisation", et les attaques différentielles et linéaires de Eli Biham, Adi Shamir et Mitsuru Matsui.

Le quatrième chapitre présente plus en détail deux exemples de réalisation pratique d'algorithmes de chiffrement qui mettent en œuvre les notions de preuve de sécurité : CS-Cipher et DFC. CS-Cipher a été développé dans un but industriel pour assurer un haut débit de chiffrement avec la technologie actuelle et offre une sécurité heuristique. En revanche, DFC est un prototype dédié à la technologie de demain, pour lequel on peut prouver une forme de sécurité. Ce dernier algorithme a été soumis au processus AES qui aboutira au standard de chiffrement du XXIème siècle.

La seconde partie de ce mémoire est une copie *in extenso* de onze de mes publications (en anglais) regroupées en annexes. Certaines sont mentionnées dans la première partie. Deux de ces articles sont parus dans le *Journal of Cryptology* et deux autres sont en cours de soumission. (Ils ont toutefois été publiés sous d'autres versions dans des actes de colloques.) Cinq autres articles sont parus dans différents volumes de la collection LNCS[7] de Springer-Verlag (ce sont les actes de Crypto, Eurocrypt, ou *Fast Software Encryption*). Un autre est publié dans les actes d'un colloque organisé par l'ACM[8]. Enfin, un dernier article a été présenté à la conférence AES organisée

[7] *Lecture Notes in Computer Sciences.*
[8] *Association for Computing Machinery.*

par la Chambre de Commerce américaine. Ces articles sont les suivants.

A. Une attaque dédiée à la fonction de hachage FFT Hash II de Claus Schnorr (Crypto 92).

B. Une attaque de la fonction de chiffrement Blowfish de Bruce Schneier (*Fast Software Encryption 96*).

C. Une méthode de cryptanalyse statistique générale qui améliore sensiblement la meilleure attaque connue contre DES (conférence ACM CCS 96).

D. La notion de "multipermutation", qui définit un critère de solidité des algorithmes (*Fast Software Encryption 94*). Cette idée provient directement de l'expérience de deux analyses : une attaque d'une version simplifiée de la fonction de hachage MD4 de Ron Rivest, et l'attaque "manquée" de la fonction de chiffrement Safer de Jim Massey.

E. Une étude sur un modèle d'attaque générique des primitives cryptographiques (travail effectué en collaboration avec Claus Schnorr qui fait la synthèse de deux articles présentés à *Fast Software Encryption* 93 et à Eurocrypt 94 sous la forme d'un article dans le *Journal of Cryptology*).

F. La notion de "décorrélation" (présentée sous différents aspects aux colloques Stacs 98 et Sac 98, et soumis au *Journal of Cryptology*).

G. La fonction CS-Cipher (travail effectué en collaboration avec Jacques Stern et publié dans *Fast Software Encryption 98*).

H. Le candidat DFC qui répond à l'appel d'offre de la Chambre de Commerce américaine pour définir un nouveau standard de chiffrement symétrique AES. Cet algorithme est issu du groupe de travail que j'ai coordonné au sein de l'Ecole Normale Supérieure.

I. Mon attaque du procédé de chiffrement à clef publique de Benny Chor et Ron Rivest (présenté à Crypto 98 et soumis au *Journal of Cryptology*).

J. L'attaque des algorithmes de signature de Adi Shamir fondés sur les permutations birationnelles effectuée par Don Coppersmith, Jacques Stern et moi-même (présenté à Crypto 93, puis publiée au *Journal of Cryptology*).

K. Mon attaque contre le standard de signature DSS (Crypto 96).

Je tiens à exprimer ma gratitude envers toutes les personnes qui m'ont accompagné de près ou de loin dans mes travaux de recherche tout au long de mes études. Dans l'exercice difficile qui consiste à les nommer en essayant de n'oublier personne, je mentionne tout d'abord Jacques Stern, qui, non content d'avoir dirigé ma thèse, m'a accompagné jusqu'à cette habilitation à diriger des recherches. C'était pour moi une chance et un privilège, et je serais surpris que notre collaboration en reste là.

Je remercie profondément Gilles Kahn qui me fait l'honneur d'accepter de présider le jury pour une habilitation sur un sujet dont il n'est pas familier, ainsi que Jean Vuillemin pour sa participation enthousiaste dans le jury. Je remercie les rapporteurs : Daniel Krob, pour qui le sujet n'est pas familier, Jean-Jacques Quisquater et Claus Schnorr qui ont à nouveau accepté d'être rapporteurs (rôle qu'ils avaient déjà accepté pour ma thèse de doctorat). Je suis particulièrement reconnaissant au fait qu'ils ont accepté ce travail en respectant des délais administratifs difficiles. A ce propos, je tiens à remercier le président du conseil scientifique de l'UFR d'Informatique de l'Université Denis Diderot, Guy Cousineau, pour avoir suivi de près ce problème de délai, ainsi que son secrétariat, et notamment Dominique Raharijesi. Un grand merci aussi au Département de Mathématiques et d'Informatique de l'Ecole Normale Supérieure pour son soutien, et notamment à Bénédicte Auffray et Joëlle Isnard.

Je remercie les personnes qui m'ont accueilli dans leur laboratoire au cours de mes recherches, soit, notamment, par ordre chronologique (depuis 1996), Ueli Maurer, Ross Anderson, Tsutomu Matsumoto, Rafael Hirschfeld, Andrew Odlyzko, Tatsuaki Okamoto.

Je suis redevable aux personnes qui m'ont fait part de leurs critiques sur le présent document, dont Louis Granboulan, David Pointcheval, et Guillaume Poupard, et avec une mention spéciale pour mon épouse Christine pour en outre les nombreux week-ends sacrifiés et son soutien permanent. Je remercie également mes autres collègues de l'Ecole Normale Supérieure, Olivier Baudron, Pierre-Alain Fouque, Philippe Hoogvorst, Phong Nguyen, Fabrice Noilhan et Thomas Pornin. J'ai été ravi de collaborer avec des partenaires industriels, notamment la Compagnie des Signaux, le Groupe Gemplus, France Telecom et d'autres qui souhaitent rester anonymes. Je remercie donc vivement Yazid Sabeg, Marc Milan, et Daniel Sabban pour la Compagnie des Signaux, David Naccache et David M'Raïhi de Gemplus, Henri Gilbert et Marc Girault de France Telecom, sans oublier le personnel du SCSSI et du Celar avec qui notre goupe entretient des liens privilégiés. Je remercie mes nombreux coauteurs. La plupart ayant déjà été mentionnés, je remercie particulièrement Florent Chabaud, Don Coppersmith, Mike Just, Kaisa Nyberg, Pascal Paillier, Bart Preneel et Dan Raphaeli. J'ajoute enfin mes remercie-

ments aux autres chercheurs avec qui j'ai collaboré : François Bayen, Eli Biham, Paul Camion, Lars Knudsen, Evangelos Kranakis, James Massey, Mitsuru Matsui, Jacques Patarin et Adi Shamir.

Je suis heureux de participer à la recherche dans un domaine en plein essor fortement concurrentiel, mais dominé par une ambiance étonnamment cordiale et qui permet de dresser un réseau d'amitié tout autour du monde.

Serge Vaudenay
Paris, Décembre 1998

Chapitre 1

Architecture

> *"Two methods [...] suggest themselves for frustrating a statistical analysis. These we may call the methods of* diffusion *and* confusion. *In the method of diffusion the statistical structure of M [the plaintext] which leads to its redundancy is "dissipated" into a long range statistics—i.e., into statistical structure involving long combinations of letters in the cryptogram. The effect here is that the enemy must intercept a tremendous amount of material to ties down this structure [...]*
>
> *The method of* confusion *is to make the relation between the simple statistics of E [the ciphertext] and the simple description of K [the secret key] a very complex and involved one."*
> Claude E. Shannon
> In *Communication Theory of Secrecy Systems*, 1949.

Ce chapitre présente les bases de la notion de chiffrement par blocs. Tout d'abord, on motive et l'on définit cette notion dans sa conception moderne. On présente ensuite les deux types d'architecture les plus répandus : la structure du schéma de Feistel et la structure de réseau de substitutions-permutations.[1]

[1] Pour un survol plus complet sur l'état de l'art dans l'architecture du chiffrement par blocs, on pourra se référer au livre de Schneier [138] ou au chapitre 7 du manuel de Menezes, van Oorschot et Vanstone [101].

1.1 Principes du Chiffrement

On présente dans cette section la problématique de la notion de chiffrement par blocs.[2]

1.1.1 Le Problème de la Confidentialité

L'objectif du chiffrement est de résoudre le problème de la "confidentialité" d'informations numériques : on souhaite rendre une donnée inintelligible à toute personne non habilitée. Ce problème intervient aussi bien dans le cas d'une transmission au travers d'un canal non protégé que dans le problème de l'archivage. Dans ce mémoire, cette information numérique est appelée "texte clair".

Un algorithme de chiffrement consiste à transformer un texte clair en un "texte chiffré" par un mécanisme de "chiffrement" de telle sorte que les personnes habilitées peuvent retrouver le texte clair en effectuant l'opération inverse de "déchiffrement". Avec des notations mathématiques, si l'on note x le texte clair, C le procédé de chiffrement (non nécessairement déterministe), et $y = C(x)$ le texte chiffré, on a une fonction de déchiffrement C^{-1} (fonction qui est nécessairement déterministe, cette fois) telle que $C^{-1}(y) = x$. Les spécifications de l'algorithme sont donc

1. C et C^{-1} sont "faciles" à réaliser, et pour chaque texte clair valide x, on a $C^{-1}(C(x)) = x$;

2. sans connaître la clef de l'algorithme, le déchiffrement est "trop difficile" à réaliser.

La notion de "difficulté" est ici purement naïve : on souhaite que le procédé puisse être raisonnablement mis en œuvre et que les personnes non habilitées ne puissent pas être capables de "décrypter" les textes chiffrés. Ce point important sera clarifié dans la définition du modèle de sécurité.

1.1.2 Principe de Kerckhoffs

Construire un algorithme de chiffrement-déchiffrement est une tâche délicate, y compris pour les experts. Comme tout le monde, *a priori*, a droit à la confidentialité, il parait difficile que les experts construisent un algorithme pour chaque personne. De plus, pour des raisons de standardisation et pour

[2]Pour une étude épistémologique sur l'évolution de la cryptologie, et notamment l'origine de la cryptographie asymétrique, consulter l'ouvrage de Stern "la science du secret" [153].

limiter les réalisations multiples, il vaut mieux privilégier la réalisation d'un algorithme unique paramétrable en fonction des utilisateurs : toute personne cherchant la confidentialité détermine son propre paramètre que l'on appelle "clef secrète"[3]. En fait, la problématique du chiffrement à notre époque est bien différente de ce qu'elle était avant la révolution industrielle. On utilise une "cryptographie de masse" : des procédés de chiffrement standardisés et massivement diffusés. Pour cette raison, il est dangereux de fonder la sécurité du procédé sur le secret des spécifications de l'algorithme.

Dans la vie courante, un mécanisme de chiffrement-déchiffrement est *a priori* connu d'un grand nombre de personnes : son inventeur, le programmeur, le vendeur du produit, ... Il peut être retrouvé par inspection du produit, divulgué par erreur ou malveillance ou tout simplement volé. L'un des principes de Kerckhoffs est que la sécurité du système ne doit jamais être fondée sur son caractère secret. En fait, dans toute étude de la sécurité procurée, il faut supposer *a priori* que l'adversaire connaît déjà tous les rouages du mécanisme.[4]

Lorsque Kerckhoffs énonça ses principes au siècle dernier dans son fameux livre *La Cryptographie Militaire*[5], il pensait en fait à l'une de ses premières applications massives : le télégraphe.

1.1.3 Chiffrement de Vernam

L'un des procédés de chiffrement fondamentaux fut inventé par Vernam à la fin de la première guerre mondiale.[6] Il mit au point un dispositif pouvant être connecté à un télégraphe, et qui permettait de chiffrer et de déchiffrer

[3]Certains auteurs préfèrent l'appeler "clef privée" pour la raison qu'elle doit être connue du chiffreur et du déchiffreur, et réserver le terme de "clef secrète" dans le cadre de la cryptographie asymétrique, où la clef est seulement connue du déchiffreur. Cependant, d'autres auteurs préfèrent réserver le terme de "clef privée" à la cryptographie asymétrique pour à peu près les mêmes raisons...

[4]Ce principe ne signifie pas, en revanche, que tout nouvel algorithme doive être publié. Ce problème est d'ailleurs sujet à controverses.

[5]Le livre de Kerckhoffs "la cryptographie militaire" [73] fut publié en 1883. (Voir Kahn [70] pour plus ample information.)

[6]A cette époque, son auteur tenta de le rendre utilisable pour des besoins militaires, mais avec la fin de la guerre, le projet tomba en désuétude, jusqu'à sa publication en 1926 [181]. (Voir Kahn [70] pour plus ample information.) Cette méthode de chiffrement coûteuse fut utilisée dans le "téléphone rouge" pendant la guerre froide. En effet, comme c'est le seul procédé de chiffrement qui soit parfaitement sûr, c'était le seul susceptible de recevoir l'accord simultané de Washington et de Moscou à cette époque où les deux pays rivalisaient de technologie pour la conquête spatiale et la course à l'armement.

les flots de données fondées sur le codage de Baudot[7].

Le procédé de Vernam consiste à traiter "au vol" le texte clair comme un flot de données en parallèle avec un flot de clefs aléatoires. Si l'on note $x = (x_1, x_2, \ldots)$ le flot de message clair et $k = (k_1, k_2, \ldots)$ le flot de clefs (ici, les x_i et k_i représentent chacun un bit, soit le chiffre 0 ou 1), le flot de message chiffré est

$$C_k(x) = ((x_1 \oplus k_1), (x_2 \oplus k_2), \ldots)$$

où $\oplus$ représente l'opération de *ou* exclusif : $0 \oplus 0 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$ et $1 \oplus 1 = 0$.[8] Ce système impose que la clef soit parfaitement aléatoire, et utilisée une unique fois. Au moment où il a été proposé, on pensait qu'il offrait une parfaite sécurité, ce qui était le cas comme cela a été démontré par Shannon plusieurs années plus tard avec le concept d'entropie.

Mais les travaux de Shannon démontrent qu'une sécurité parfaite impose les règles suivantes.

- La clef doit avoir la même longueur que le texte clair. Il faut donc connaître *a priori* la longueur du texte avant de le transmettre pour synchroniser l'émetteur et le récepteur sur une même clef. La clef ne doit ensuite jamais être réutilisée.

- La clef doit être parfaitement aléatoire. C'est dans la pratique difficile à obtenir pour des raisons de physique théorique.[9]

L'existence d'un système parfaitement sûr ne signifie donc pas l'arrêt des travaux de recherche en cryptographie symétrique car un tel système est nécessairement inadapté aux applications industrielles usuelles. En fait, c'est la sécurité parfaite dans le modèle de Shannon qui est trop restrictive. Trouver un système plus flexible et mieux adapté aux contraintes de sécurité pratique est donc le défi de la recherche actuelle en cryptographie.

1.1.4 Chiffrement par Blocs et Chiffrement en Chaîne

La notion de chiffrement doit surmonter un problème évident : l'ensemble des textes clairs est *a priori* infini. En fait, en accord avec la conception traditionnelle de l'informatique, cet ensemble, que l'on appelle "espace de

[7]Le codage de Baudot était utilisé pour les téléscripteurs. Contrairement au codage de Morse, il utilise un alphabet binaire, et chaque caractère alphabétique est codé par un mot de longueur fixe sur cinq bits.

[8]En fait, le procédé original de Vernam utilisait l'opération complémentaire, ce qui est strictement équivalent.

[9]Les lois de la thermodynamique.

messages", est l'ensemble des mots sur l'alphabet binaire $\{0, 1\}$. Autrement dit, on considère un message comme une suite finie de 0 et de 1, soit une *chaîne de bits*. Un message de longueur m s'écrit donc $x = (x_1, x_2, \ldots, x_m)$ où $x_i = 0$ ou 1. On doit donc définir un algorithme sur un tel ensemble infini.

Le chiffrement de Vernam est un exemple de "chiffrement en chaîne" : l'algorithme de chiffrement traite le flot de texte clair "au vol" et produit un flot de texte chiffré. On compare ce principe à la notion de "chiffrement par bloc" pour lequel le texte clair est préalablement découpé en "blocs de message" qui sont traités séparément. Comme la plupart des microprocesseurs traitent des mots de plusieurs bits (de 32 bits, souvent), cette opération s'avère rentable une fois mise en œuvre. Dans la pratique, on utilise des modes de chiffrement hybrides : on considère le texte clair comme un flot de blocs de message qui sont traités "au vol". On appelle "mode opératoire" ce mécanisme de traitement à partir d'une fonction de chiffrement par blocs.[10]

1.1.5 Mode Opératoire

On suppose que l'on a défini une "primitive de chiffrement" admettant en entrée un bloc de message x de longueur fixe m et une clef k et fournissant un bloc de message chiffré $y = C_k(x)$ de longueur m. Plusieurs modes opératoires permettent d'adapter cette primitive pour définir un algorithme de chiffrement. On décrit deux modes opératoires courants.[11]

Mode ECB Suivant le mode ECB[12], le texte clair est découpé en blocs de longueur m, soit $x = (x_1, \ldots, x_n)$. Le texte chiffré est $y = (y_1, \ldots, y_n)$ où $y_i = C_k(x_i)$. Le déchiffrement est évident.

Mode CBC Suivant le mode CBC[13], on définit un "bloc initial" y_0, et le texte clair est découpé en blocs de longueur m, soit $x = (x_1, \ldots, x_n)$. Le texte chiffré est $y = (y_1, \ldots, y_n)$ où $y_i = C_k(x_i \oplus y_{i-1})$. Pour déchiffrer, on calcule itérativement $x_i = C_k^{-1}(y_i) \oplus y_{i-1}$.

L'existence de ces modes de chiffrement permet de se concentrer sur la primitive de chiffrement, c'est-à-dire sur la fonction qui permet de chiffrer un

[10] Le présent mémoire ne traite pas du chiffrement en chaîne. Pour une étude sur le sujet, voir l'ouvrage de Rueppel [134] ou l'article de synthèse [135].

[11] Les modes opératoires ECB et CBC ont été normalisés successivement par le "National Bureau of Standards" (NBS) [3] (qui était une partie du ministère de l'industrie des Etats Unis d'Amérique avant la création du "National Institute of Standards and Technology" (NIST)), l'"American National Standards Institute" (ANSI) [1] et par l'"International Organization for Standardization" (ISO) [7, 8].

[12] De l'anglais *Electronic Code Book*.

[13] De l'anglais *Cipher Block Chaining*.

bloc.

1.1.6 Algorithme de Chiffrement Symétrique Formel

On rappelle les principes de construction des algorithmes de chiffrement qui ont été discutés dans les précédentes sections.

- L'algorithme doit être symétrique, donc le déchiffrement s'obtient par transformation simple à partir du chiffrement.
- L'algorithme doit être public suivant les principes de Kerckhoffs, donc utiliser une clef secrètement déterminée par l'utilisateur.
- L'algorithme doit reposer sur une primitive de chiffrement, c'est-à-dire traiter des blocs de message de longueur fixée et s'étendre aux flots de textes clairs par un mode opératoire.
- L'algorithme ne doit pas être trop coûteux en terme de mise en œuvre.
- L'algorithme doit offrir une réelle sécurité en pratique. (Ce dernier point sera précisé dans le chapitre 3.)

Les primitives de chiffrement peuvent habituellement se décrire suivant plusieurs méthodes complémentaires.

- Par un programme. Il est assez naturel de considérer le mécanisme de chiffrement comme un "automate fini déterministe". Des registres internes à l'automate définissent son "état", défini au départ par le bloc de message à chiffrer et la clef secrète. Une succession d'étapes de calculs modifie ensuite cet état suivant des règles de transitions. L'état final de l'automate définit le bloc de message chiffré.
- Par un "réseau de calcul". En général, les étapes de calculs de l'automate (et en particulier le nombre de ces étapes) sont indépendantes du message à chiffrer. Cela permet de les décrire par un diagramme qui illustre le procédé.

Cette notion de réseau de calcul se prête plus facilement à un traitement théorique utile. (Si l'on conservait la notion plus générale de programme, on aurait à faire face, notamment, au problème de l'arrêt.) On définit donc formellement la notion de réseau de calcul qui sera utilisée dans la suite. Pour cela, on rappelle quelques définitions et notations usuelles.

Définition 1.1. *On appelle graphe orienté la donnée d'un couple (V, E) formé d'un ensemble V et d'une partie E de V^2. V est l'ensemble des "sommets", et E est l'ensemble des "arêtes". Une arête $a = (u, v)$ a pour "origine" u et "extrémité" v. Pour un sommet u, on note In_u l'ensemble des arêtes d'extrémité u et Out_u l'ensemble des arêtes d'origine u. Un cycle est une suite d'arêtes $a_1, \ldots, a_n$ telle que l'extrémité de a_i (respectivement a_n) est l'origine de a_{i+1} (respectivement a_1) pour $i = 1, \ldots, n-1$.*

On sépare les descriptions géométriques et calculatoires d'un réseau de calcul.

Définition 1.2. *Un "réseau de calcul" est la donnée $G = (V, E, \mathrm{Dom})$ d'un graphe orienté sans cycle (V, E) dans lequel chaque arête a est étiquetée par un ensemble Dom_a. On distingue l'ensemble des "sommets d'entrée" In_G (les sommets qui ne sont l'extrémité d'aucune arête) et l'ensemble des "sommets de sortie" Out_G (les sommets u qui ne sont l'origine d'aucune arête). On impose en outre que les sommets d'entrée et de sortie ne soient adjacents qu'à une seule arête, et l'on notera également In_G et Out_G l'ensemble de ces arêtes, respectivement. (Lorsque cela n'introduira aucune ambiguïté de notation.)*

Définition 1.3. *Pour un réseau de calcul $G = (V, E, \mathrm{Dom})$, étant donné un ensemble A d'arêtes de G, on note*

$$\mathrm{Dom}(A) = \prod_{a \in A} \mathrm{Dom}_a.$$

On appelle "évaluation partielle" définie sur A tout élément t de $\mathrm{Dom}(A)$, et l'on note $t(a)$ la "valeur" de l'arête a, c'est-à-dire l'élément de Dom_a correspondant. On appelle "évaluation" de G toute évaluation partielle définie sur E. Pour toute évaluation partielle t de G définie sur un ensemble A, si B est un sous-ensemble de A, on note $t|_B$ l'unique évaluation partielle définie sur B qui coïncide avec t.

Définition 1.4. *Pour un réseau de calcul $G = (V, E, \mathrm{Dom})$, une "interprétation du réseau" f est un étiquetage qui à chaque sommet u qui n'est pas un sommet d'entrée ou de sortie associe une fonction f_u de $\mathrm{Dom}(\mathrm{In}_u)$ vers $\mathrm{Dom}(\mathrm{Out}_u)$.*

Intuitivement, le réseau de calcul décrit géométriquement une suite finie de calculs. Chaque arête représente un résultat intermédiaire, et chaque sommet représente une "boîte de calcul". Une interprétation du réseau décrit la définition de ces boîtes. [14]

[14] Dans une telle définition, chaque arête admet une unique extrémité, ce qui revient à dire que chaque résultat intermédiaire n'est utilisé qu'une seule fois. On pourrait étendre

Le but d'un réseau de calcul est de modéliser une fonction. On définit donc un vocabulaire approprié.

Définition 1.5. *Pour un réseau de calcul interprété* $G = (V, E, \mathrm{Dom}, f)$*, une "entrée" est une évaluation partielle définie sur* In_G*. Une "sortie" est une évaluation partielle définie sur* Out_G*. Pour tout sommet* u*, une évaluation partielle* t *de* G *définie sur un ensemble* A *qui contient* In_u *et* Out_u *est "admissible" en* u *si l'on a*

$$t|_{\mathrm{Out}_u} = f_u\left(t|_{\mathrm{In}_u}\right).$$

Une évaluation de G *est admissible si elle l'est en tout sommet. Pour une entrée* x *de* G*, on note* $\mathrm{Val}^G(x)$ *l'unique évaluation admissible de* G *qui coïncide avec* x*, et l'on note* $G(x)$ *l'unique sortie de* G *qui coïncide avec* $\mathrm{Val}^G(x)$*.*

Avec ces définitions, on assimile ainsi un réseau de calcul interprété $G = (V, E, \mathrm{Dom}, f)$ à une fonction de $\mathrm{Dom}(\mathrm{In}_G)$ vers $\mathrm{Dom}(\mathrm{Out}_G)$.

1.2 Schéma de Feistel

Une idée simple pour définir un automate qui soit "localement réversible" consiste à calculer des "sous-clefs" $k_1, \ldots, k_r$ en fonction de la clef secrète k, à découper le bloc de message en deux parties qui occupent deux registres séparés x_1 et x_2, et à effectuer une succession de r "étages" du type

1. $x_1 \leftarrow f(x_1, g^{k_i}(x_2))$

2. échanger les registres x_1 et x_2.

(Voir Fig. 1.1.) L'échange du dernier étage est souvent omis.

Il n'y a aucune contrainte sur la fonction g^{k_i} que l'on appelle "fonction d'étage". En revanche, afin d'assurer la réversibilité locale, la fonction f doit être telle que si $f(x_1, u) = f(x_1', u)$, alors on a nécessairement $x_1 = x_1'$. Ainsi $f(., u)$ doit-elle être une permutation pour tout u. On utilise pour cela une loi de groupe.

---

la définition à la notion d'"hypergraphe", mais cela rendrait l'illustration graphique plus difficile. On privilégiera donc l'adjonction de "fausses-boîtes de calcul" dont le but est de dupliquer les données : ce sont des sommets u tels que In_u est réduit à une seule arête a et tels que $f_u(t)(b) = t(a)$ pour tout $t \in \mathrm{Out}_u$.

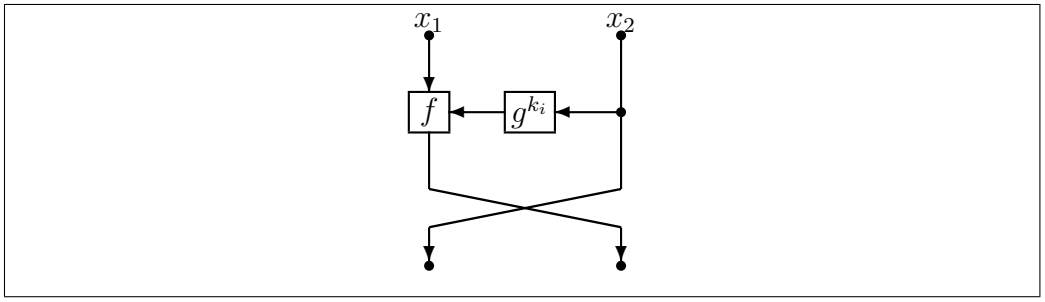


Figure 1.1 – Etage du Schéma de Feistel.

1.2.1 Description Fonctionnelle

Cette idée est à l'origine du "schéma de Feistel", du nom de l'inventeur du l'algorithme de chiffrement Lucifer publié en 1973.[15] Soient $\mathcal{M} = \mathcal{M}_0^2$ et $\mathcal{M}_0 = \mathbf{Z}_2^{\frac{m}{2}}$ où m est la longueur des blocs de message à chiffrer. L'ensemble $\mathcal{M}_0$ est naturellement muni d'une loi de groupe que l'on note $\oplus$ définie par

$$\left(b_1, \ldots, b_{\frac{m}{2}}\right) \oplus \left(b'_1, \ldots, b'_{\frac{m}{2}}\right) = \left(b_1 + b'_1 \bmod 2, \ldots, b_{\frac{m}{2}} + b'_{\frac{m}{2}} \bmod 2\right).$$

C'est le *ou* exclusif bit-à-bit. On note que la "soustraction" correspondante coïncide avec l'"addition" puisque l'on a $(a \oplus b) \oplus b = a$ pour tout a et b. La fonction f est donc en fait $f(x_1, u) = x_1 \oplus u$. La fonction "fonction d'étage" reste arbitraire. On note que l'étape finale d'échange des deux registres est souvent omise. Cela permet de conserver la même structure pour l'opération de déchiffrement : on applique le même procédé en inversant l'ordre des sous-clefs. Tout au long de ce mémoire, on utilise la notation suivante.

Définition 1.6. *Soit m un entier naturel pair. Soit le groupe $\mathcal{M}_0 = \mathbf{Z}_2^{\frac{m}{2}}$ et l'espace de blocs message $\mathcal{M} = \mathcal{M}_0^2$. Etant donnés un entier naturel r et r fonctions $g_1, \ldots, g_r$ sur $\mathcal{M}_0$, on définit la permutation $\Psi(g_1, \ldots, g_r)$ sur $\mathcal{M}$ par le procédé de la figure 1.2.*

Si l'on a une famille de fonctions $g^{k_i} : \mathcal{M}_0 \to \mathcal{M}_0$ définies par un paramètre k_i, on définit $g_i(x) = g^{k_i}(x)$. On note alors que $\Psi(g^{k_1}, \ldots, g^{k_r})$ est bien une permutation sur $\mathcal{M}$ et que sa permutation inverse est $\Psi(g^{k_r}, \ldots, g^{k_1})$, ce qui facilite grandement la mise en œuvre.

[15] Le schéma de Feistel a été défini dans un article de synthèse [51] publié en 1973. Il apparaît dans la fonction de chiffrement Lucifer qui a été proposée comme standard de chiffrement. Le National Bureau of Standard a demandé une nouvelle version de ce procédé à IBM, ce qui a conduit au standard DES. (Pour plus de précisions, voir l'article de synthèse de Smid et Branstad [152].) La sécurité de la fonction Lucifer a été par la suite analysée par Ben-Aroya et Biham [22].

Entrée : bloc de message clair x

Sortie : bloc de message chiffré y

1. $(x_1, x_2) \leftarrow x$
2. $x_1 \leftarrow x_1 \oplus g_1(x_2)$
3. échange x_1 et x_2
4. $x_1 \leftarrow x_1 \oplus g_2(x_2)$
5. échange x_1 et x_2
6. ...
7. $x_1 \leftarrow x_1 \oplus g_r(x_2)$
8. $y \leftarrow (x_1, x_2)$

FIGURE 1.2 – Fonction $\Psi(g_1, g_2, \ldots, g_r)$.

En général, un algorithme de chiffrement fondé sur le schéma de Feistel est caractérisé par

- un nombre d'étages r ;
- un algorithme de diversification de clefs qui à toute clef k associe r sous-clefs $k_1, \ldots, k_r$;
- une famille de fonctions qui à toute sous-clef k_i associe une fonction g^{k_i}.

La primitive de chiffrement est alors $C = \Psi(g^{k_1}, \ldots, g^{k_r})$, et la primitive de déchiffrement est $C^{-1} = \Psi(g^{k_r}, \ldots, g^{k_1})$. Par exemple, la figure 1.3 représente un procédé de chiffrement de Feistel à trois étages dans lequel la clef est diversifiée en trois sous-clefs k_1, k_2 et k_3.

Pour décrire la structure de $\Psi(g^{k_1}, g^{k_2}, g^{k_3})$ suivant le formalisme de réseau de calcul, on définit le graphe orienté $G = (V, E)$ par

$$\begin{aligned} V &= \{u_1, \ldots, u_6, v_1, \ldots, v_7, w_1, w_2, w_3\} \\ E &= \{(v_1, u_4), (v_2, w_1), (v_3, u_1), (w_1, u_1), (u_1, u_4), (u_4, w_2), (w_1, u_5), \ldots\} \end{aligned}$$

dans lequel toutes les arêtes étiquetées par le même ensemble $\mathbf{Z}_2^{\frac{m}{2}}$ (voir Fig. 1.4). Dans ce réseau, les sommets d'entrée sont $v_1, \ldots, v_5$, les sommets de sortie sont v_6 et v_7. Les sommets w_1, w_2 et w_3 représentent des "fausses-boîtes de calcul" dont le but est de dupliquer les résultats provenant de v_2,

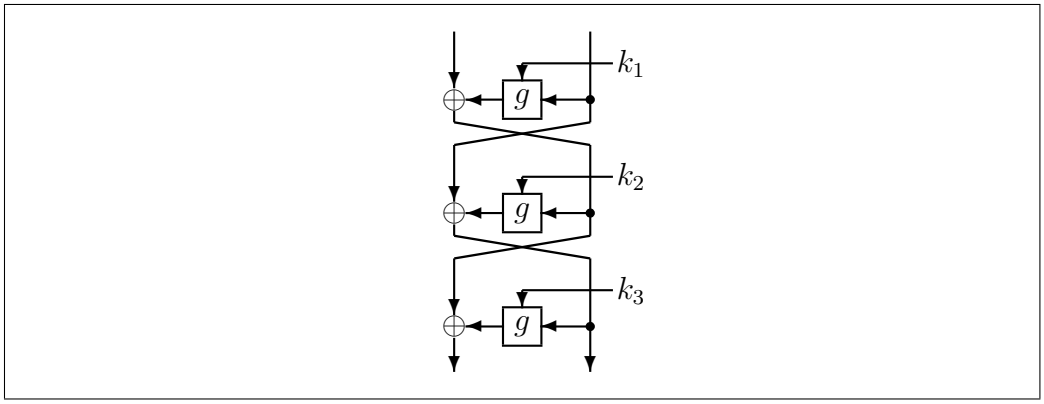


Figure 1.3 – Fonction $\Psi(g^{k_1}, g^{k_2}, g^{k_3})$.

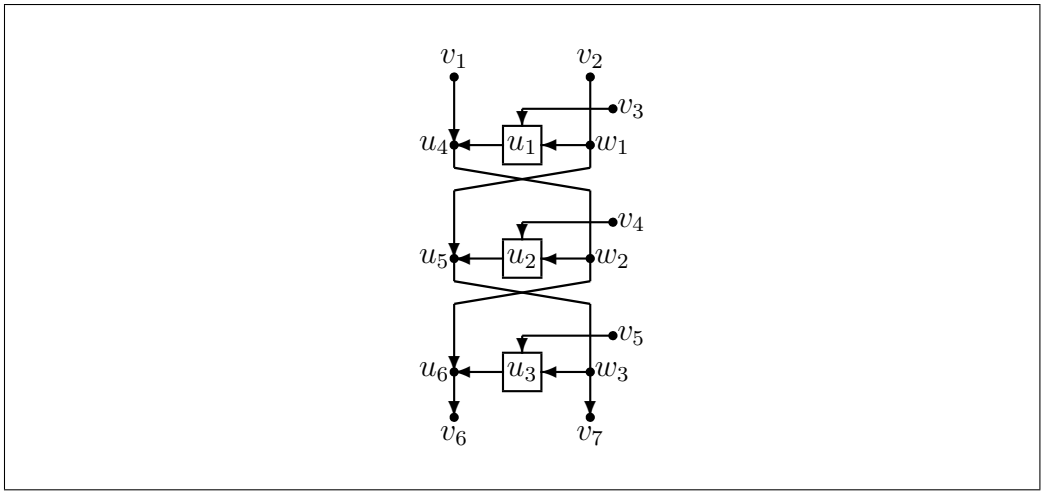


Figure 1.4 – Réseau de Calcul de $\Psi(g^{k_1}, g^{k_2}, g^{k_3})$.

u_4 et u_5 respectivement. Les sommets u_4, u_5 et u_6 s'interprètent en des fonctions $\oplus$, et les sommets u_1, u_2 et u_3 s'interprètent en une fonction g. Dans la suite, on représentera directement le diagramme de la figure 1.3 qui est plus parlant et la notion du réseau de calcul sous-jacente sera laissée au lecteur.

1.2.2 Généralisations Possibles

On généralise facilement la notion de schéma de Feistel de plusieurs façons.[16]

Nouvelles Permutations : on peut insérer des permutations fixes entre chaque étape. Par exemple, on peut insérer une permutation initiale du

[16]Pour un survol des généralisations possibles du schéma de Feistel, voir Schneier et Kelsey [139].

bloc de message clair x et une permutation finale du bloc de message chiffré y. C'est le cas, par exemple, du standard DES.

Autre structure : on peut remplacer la structure du groupe $\mathbf{Z}_2^{\frac{m}{2}}$ par tout ensemble $\mathcal{M}_0$ muni d'une loi d'action f d'un ensemble $\mathcal{A}$ sur $\mathcal{M}_0$: une fonction $f : \mathcal{M}_0 \times \mathcal{A} \to \mathcal{M}_0$ telle que $f(., u)$ est une permutation de $\mathcal{M}_0$ pour tout u. La fonction d'étage g est alors à valeur dans $\mathcal{A}$. Plus simplement, on peut remplacer la loi $\oplus$ par toute autre loi de groupe, ou de pseudo-groupe.

Diversification des étages : la fonction d'étage, ainsi que la structure utilisée, peut être modifiée d'un étage à l'autre.

Schéma non équilibré : il est possible d'utiliser des registres qui évoluent dans des ensembles différents, soit $\mathcal{M} = \mathcal{M}_1 \times \mathcal{M}_2$. Dans ce cas, il est nécessaire d'alterner la structure des étages. Pour un étage sur deux, on utilise une action f_1 sur $\mathcal{M}_1$ et une fonction g_1 partant de $\mathcal{M}_2$, et dans les autres étages, on utilise une action f_2 sur $\mathcal{M}_2$ et une fonction g_2 partant de $\mathcal{M}_1$.

Extension du nombre de registres : on peut évidemment utiliser plus de deux registres. Par exemple, les fonctions de hachage MD4 et MD5 de Rivest utilisent quatre registres. La figure 1.5 illustre un exemple de schéma de Feistel à quatre registres. De même, le standard de hachage SHA utilise cinq registres.[17]

1.3 Réseaux de Substitutions-Permutations

Le formalisme de réseau de calcul permet de discuter des concepts de "diffusion" et de "confusion" définis par Shannon. La confusion consiste à transformer une fraction d'information pour la rendre inintelligible. On peut dire, par exemple, que le rôle des boîtes à une seule entrée consiste à réaliser cette opération. La diffusion, en revanche, consiste à répartir une fraction d'information dans plusieurs positions. Afin de réaliser la notion de réversibilité locale il est nécessaire que les boîtes qui réalisent cette opération aient plusieurs

[17] Les fonctions de hachage MD4 et MD5 ont été proposées par Rivest comme standard à la communauté Internet. Elles ont été publiées dans [128, 129, 130], et [131]. La fonction de hachage SHA est normalisée par la "National Institute of Standards and Technology" (NIST) dans [6] et en cours de normalisation par l'"International Organizations for Standardization". La description d'origine de SHA publiée dans [4] en 1993 a été mise à jour dans [5] en 1995.

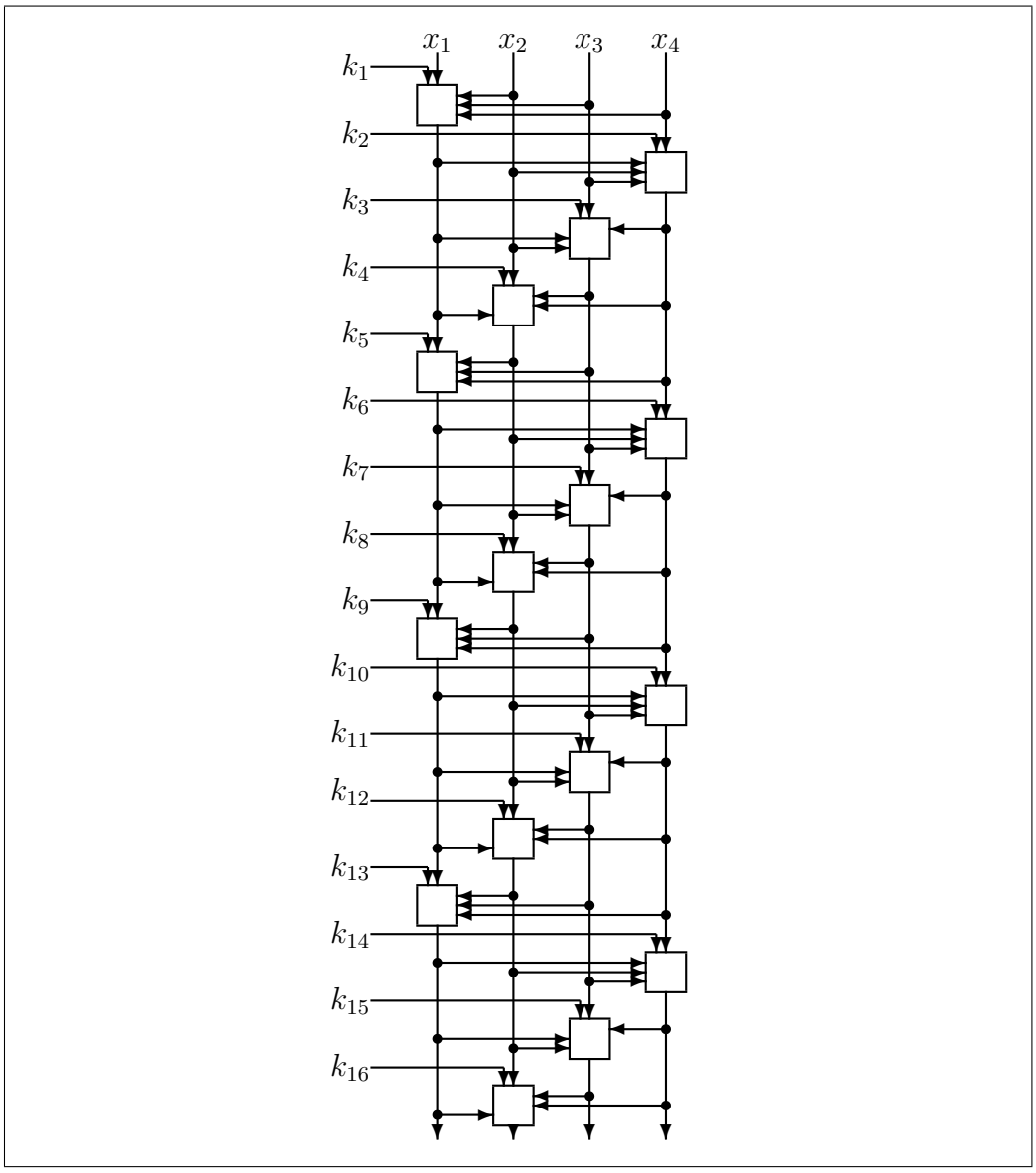


Figure 1.5 – Schéma de Feistel à Quatre Registres.

entrées. La notion de réseau de calcul permet donc de distinguer facilement les boîtes de calcul correspondantes. Suivant Shannon, diffusion et confusion sont les ingrédients essentiels pour réaliser une fonction de chiffrement.

Suivant une dichotomie analogue utilisée dans la littérature, on parle de "substitution" pour toute boîte de calcul et de "permutation" pour tout enchevêtrement d'arêtes. On appelle donc de tels réseaux de calcul des "réseau de substitutions-permutations".[18]

La difficulté de la construction réside dans la condition que la fonction doit être bijective. La méthode de Feistel illustrée par la figure 1.1 (p. 23) est une solution possible. Dans ce réseau, g^{k_i} est une fonction quelconque (donc pas nécessairement une permutation), et f est (le plus souvent) une loi de groupe. Une approche différente consiste à réaliser un réseau de calcul composé de permutations sur des petits objets. Ces objets étant petits, ces opérations se définissent facilement par des tables. Le nombre de "registres" est cependant plus grand. Cela rend peu rentable la mise en œuvre sur des microprocesseurs qui manipulent des mots de 32 ou 64 bits, mais rend efficace celle fondée sur une architecture à 8 bits. Suivant un usage courant, on parle de "réseau de substitutions-permutations" lorsque l'on n'utilise pas le schéma de Feistel. Cette utilisation impropre (puisque le schéma de Feistel est en fait un exemple de tel réseau) provient du fait que l'on n'a pas de nom générique pour ces constructions particulières.

1.3.1 Constructions Fondées sur le Graphe FFT

Une idée naturelle pour réaliser une bonne diffusion d'information sur un grand nombre de registres consiste à utiliser le graphe correspondant à l'algorithme de transformation de Fourier rapide FFT[19]. Pour cela, le nombre de registres doit être une puissance p^n d'un nombre p. Le graphe est alors composé de n étages identiques. Chaque étage comporte p^{n-1} boîtes de calcul à p entrées et p sorties.

On représente les sommets d'entrée $u_{i_1,\ldots,i_n}$ d'un étage par un vecteur de n indices $i_1, \ldots, i_n$ compris entre 0 et $p-1$. Les boîtes de calcul $v_{i_1,\ldots,i_{n-1}}$ sont repérées par $n-1$ indices. De même, les sommets de sortie sont notés $w_{i_1,\ldots,i_n}$. Par définition, les arêtes entrantes de $v_{i_1,\ldots,i_{n-1}}$ sont les arêtes provenant des $u_{i_1,\ldots,i_n}$ pour tout i_n, et les arêtes sortantes ont pour extrémité les $w_{i_n,i_1,\ldots,i_{n-1}}$ pour tout i_n. Un étage de FFT consiste donc à traiter les données par paquets de p et à faire une rotation circulaire sur les indices. (En particulier, si les

[18] La notion de réseau de substitutions-permutations a été étudiée pour la première fois par Kam et Davida [72]. Elle a été reprise et étudiée par l'"école de Tavares" : dans la thèse d'Adams [11], la thèse de Heys [63], ...

[19] De l'anglais *Fast Fourier Transform*.

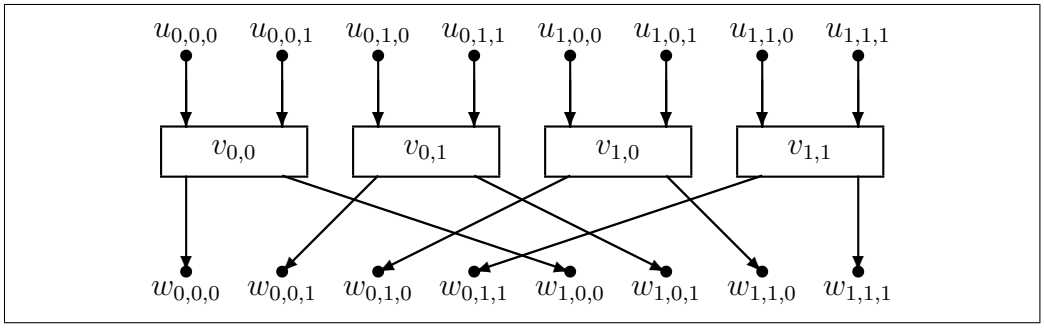


Figure 1.6 – Un Etage de FFT à 2^3 Registres.

boîtes de calcul ne font aucune opération, une succession de n tels étages ne permute donc aucune information.) La figure 1.6 représente un étage de FFT pour 2^3 registres.

Le graphe FFT a été notamment utilisé par Schnorr pour définir une fonction de hachage "FFF-Hashing"[20] et par Massey dans la fonction Safer.

1.3.2 Exemple de la Fonction Safer

La fonction Safer inventée par Massey utilise un graphe FFT à 2^3 registres. Dans sa version d'origine, cette fonction est composée de six étages.[21] Chacun de ces étages est composé d'une couche "$\oplus/+$", d'une couche de permutations spécifiques P et Q, d'une couche "$+/\oplus$", et de trois étages de FFT (voir Fig. 1.7). Chacun des registres représente un octet. Les permutations P et Q peuvent donc facilement être représentées par des tables. On a

$$P(x) = (45^x \bmod 257) \bmod 256.$$

Cette fonction est une permutation pour des raisons algébriques[22]. La fonction Q n'est autre que P^{-1}. La fonction 2PHT à deux entrées et deux sorties est purement linéaire. On a

$$\mathrm{2PHT}(x, y) = (2x + y \bmod 256, x + y \bmod 256).$$

On note que la dernière couche de permutation du graphe FFT a été supprimée ici.

[20] Voir [141].

[21] La fonction de chiffrement Safer a été publiée dans [87]. Sa sécurité a été étudiée dans Vaudenay [162] (voir annexe D), Massey [88], Berson et Knudsen [23]. Une extension a été proposée dans Knudsen [75].

[22] 257 est un nombre premier et donc le groupe $\mathbf{Z}_{257}^*$ est isomorphe à $\mathbf{Z}_{256}$. De plus, 45 est un générateur du groupe $\mathbf{Z}_{257}^*$.

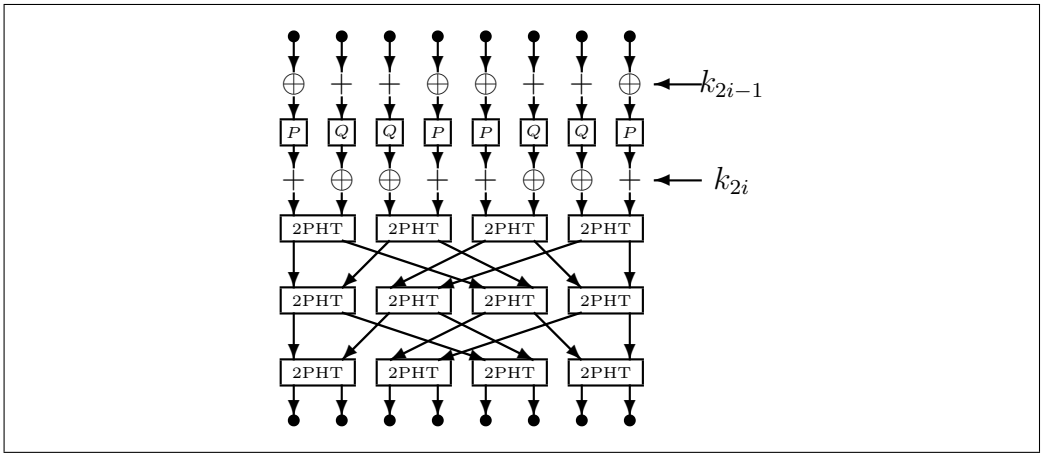


FIGURE 1.7 – Un Etage de Safer.

Dans ce procédé de chiffrement, chaque étage dispose de 16 octets de clefs qui composent les vecteurs k_{2i-1} et k_{2i}. Ces valeurs proviennent d'un mécanisme de diversification de clefs qui n'est pas détaillé ici. On note, de plus, qu'à l'issue des six étages de Safer, une couche "$\oplus/+$" avec un treizième vecteur k_{13} est ajoutée (sinon, le graphe FFT du sixième étage n'apporte rien à la sécurité).

La fonction CS-Cipher est analogue. Elle est présentée en annexe G.

1.4 Conclusion

Si l'on considère le contexte industriel actuel, la confidentialité doit se résoudre par un algorithme

- symétrique, simple et peu coûteuse, qui utilise une clef secrètement déterminée par l'utilisateur,
- que l'on peut supposer public, suivant les principes de Kerckhoffs,
- et qui repose sur une primitive de chiffrement, c'est-à-dire traiter des blocs de message de longueur fixée et s'étendre aux flots de textes clairs par un mode opératoire.

En outre, il doit offrir une réelle sécurité et être exempts d'attaques.

Les méthodes de construction classiques distinguent la construction par le schéma de Feistel (et ses généralisations) des autres "réseau de substitutions-permutations" qui sont en fait des réseaux composés de boîtes de calcul réversibles.

Cela permet de remarquer que tous ces procédés utilisent en fait une description géométrique, ce que l'on formalise par la notion de "réseau de calcul", et sur lesquels on branche des boîtes de calcul.

Chapitre 2

Attaques sur le Chiffrement par Blocs

> *"The open cryptographic literature contains very few examples of universal methods of cryptanalysis, which can be successfully applied to a wide variety of encryption and hash function."*
> Eli Biham and Adi Shamir
> In *Differential Cryptanalysis of the Data Encryption Standard*, 1993.

Dans ce chapitre, on présente quelques méthodes d'attaque classiques qui dégagent une analyse générale : il existe de nombreuses attaques dédiées à certains procédés cryptographiques et non transposables. Elles ne font pas partie de cette étude. Bien que ce chapitre se fonde sur des exemples bien précis, l'idée de chaque type d'attaques s'adapte facilement à d'autres algorithmes. Les méthodes sont illustrées par des exemples d'attaque :

- une attaque différentielle fondée sur les clefs faibles de Blowfish (qui améliore celle qui est présentée en annexe B) ;
- une attaque linéaire (manquée) de Safer (voir annexe D) ;
- une attaque par boîtes noires de FFT Hash II (voir annexe A) ;
- une attaque statistique par projection de DES (voir annexe C).

2.1 Modèles de Sécurité

Avant d'entrer dans le détail des méthodes d'attaque, il convient de cerner la notion de modèle d'attaque.[1] Que peut-on présupposer de la puissance de calcul de l'attaquant ? A quelles informations peut-il avoir accès ? Quels sont ses véritables objectifs ?

On note que l'on ne considère ici que les modèles d'attaque logiques. De nombreux modèles qui ont récemment marqué l'actualité cryptographique reposent sur des attaques physiques qui utilisent des défauts de mise en œuvre des procédés plutôt que les faiblesses intrinsèques des algorithmes. On mentionne par exemple l'attaque par mesure de temps de calcul, de consommation électrique, par provocation d'erreurs de calcul dans des conditions physiques extrêmes, par accès aux circuits câblés, ... Ces modèles n'entrent pas dans le cadre de ce mémoire.

2.1.1 Information Accessible à l'Attaquant

D'après le principe de Kerckhoffs, il n'est pas raisonnable de supposer le secret de la description de l'algorithme. On suppose donc que l'attaquant connaît la description formelle du procédé de chiffrement mais que seule la clef secrète (ou une partie) lui est cachée.

Dans un modèle d'attaque minimal, l'attaquant ne peut réaliser que des écoutes des communications chiffrées : mathématiquement, il dispose d'une source aléatoire $C_k(X)$ de textes chiffrés pour lesquels il suppose une certaine distribution des textes clairs correspondants. Il peut, par exemple, supposer que X est une lettre de type administratif rédigée en français qui commence par l'identification de l'expéditeur, du destinataire, de la date et du lieu d'émission, par "Cher Monsieur [...]", *etc.* L'hypothèse que le texte est en français révèle beaucoup d'autres d'informations sur X : que la plupart des caractères sont alphabétiques, minuscules, que le caractère le plus fréquent est "`e`", qu'une lettre presque systématiquement suivie par une même autre lettre correspondra certainement au digramme "`qu`", *etc.* On dit que ce modèle d'attaque est "à texte chiffré connu seulement".

Dans un modèle d'attaque plus fort, on suppose que l'attaquant a accès au texte clair également. Cela se justifie s'il peut "regarder par dessus l'épaule" de l'émetteur, ou s'il obtient l'accès au texte par d'autres moyens. Si par exemple une entreprise dévoile un texte contenant une information intéres-

[1] Comme l'analyse cryptographique présente alternativement deux points de vue opposés, celui de l'attaquant qui cherche à casser un système, et celui d'un expert qui cherche à en prouver la sécurité, la notion de modèle d'attaque est identique à celle de modèle de sécurité.

sante pour ses concurrentes, ce document sera certainement retransmis dans une communication chiffrée entre deux filiales d'une autre entreprise. Suivant un autre scénario, le récepteur peut être tout simplement peu consciencieux et omettre de protéger le texte déchiffré. Ce modèle d'attaque dit "à texte clair connu" est donc justifié dans de nombreux contextes.

Un attaquant encore plus fort peut également choisir le texte clair pour observer le résultat chiffré. Par exemple, dans le cas où le dispositif de chiffrement est scellé et où l'attaquant y a accès pendant une période limitée, il peut librement faire des expériences dans le but d'attaquer le système. C'est un modèle "à texte clair choisi".[2]

Il existe d'autres modèles.

- Les attaques "à texte chiffré choisi". Mais ce modèle est surtout utilisé dans le contexte de la cryptographie asymétrique (car pour le chiffrement symétrique, ce modèle est souvent semblable à celui des attaques à texte chair choisi). [3]

- Les attaques à clefs liées. Dans ce modèle, on suppose que l'on a plusieurs dispositifs de chiffrement qui utilisent des clefs liées par certaines propriétés. (Par exemple un utilisateur change son mot de passe mais en choisit un peu différent du précédent.)[4]

- Les attaques par clefs faibles. On suppose ici que l'on cherche à attaquer un système de chiffrement particulier au sein d'un réseau, mais sans savoir lequel *a priori*. Ce système se caractérise en ce qu'il utilise une clef réputée faible. L'attaquant doit donc préalablement le détecter et isoler ce système des autres.

2.1.2 Puissance de l'Attaquant

La puissance de calcul de l'attaquant est un paramètre important du modèle d'attaque. Dans le cas où sa puissance est illimitée, on s'intéresse à la capacité d'attaquer à partir des informations disponibles. Suivant les auteurs, on parle de "modèle de la théorie de l'information", de "modèle de Shannon", ou de "modèle de sécurité parfaite".

[2]Les types d'attaques à texte clair connu ou choisi ou à texte chiffré seulement sont exposés dans tous les ouvrages de cryptographie. Par exemple, le livre de Stinson [156].

[3]On appelle parfois "attaques du déjeuner" ce modèle, car on imagine que l'attaquant peut librement accéder au système de déchiffrement pendant que ses responsables sont partis déjeuner...

[4]La notion d'attaque à clefs reliées est due à Biham [25].

La cryptographie asymétrique utilise beaucoup de modèles où les attaquants sont supposés incapables de résoudre des problèmes particuliers comme le problème de la factorisation, du logarithme discret, ou encore des problèmes NP-complets. On s'intéresse donc naturellement aux attaquants limités par un temps de calcul polynomial. Comme les paramètres utilisés en cryptographie symétrique sont moins variables que ceux de la cryptographie asymétrique, on peut cependant s'interroger sur le sens du "temps de calcul polynomial". On préfère donc limiter la puissance de calcul des attaquants en nombre d'opérations élémentaires et en nombre d'unités de mémoire. De même, on s'intéresse à la probabilité de succès de l'attaque.

Dans les modèles d'attaque, les paramètres qui caractérisent la puissance de l'attaquant sont donc

- la complexité en nombre d'instructions élémentaires ;
- la complexité en nombre de bits de mémoire utilisés ;
- la probabilité de succès.

2.1.3 Objectifs de l'Attaquant

Le but ultime d'un attaquant est de décrypter un message auquel il n'a pas réussi à avoir accès physiquement. En effet, dans les modèles à textes clairs connus, il obtient l'accès par une faille indépendante de l'algorithme de chiffrement. L'objectif de ce mémoire est l'étude de la sécurité apportée par l'algorithme même de chiffrement, il convient donc de s'intéresser à la protection des messages qui n'ont pas été dévoilés par d'autres moyens.

Un objectif intermédiaire consiste à obtenir la clef secrète. Il est clair qu'elle permet de décrypter tout nouveau message. En revanche, il est possible qu'une attaque qui parvient à décrypter un message donné ne permette pas pour autant l'accès à la clef. Le modèle de sécurité contre les attaques par décryptage est donc plus fort que le modèle de sécurité contre les attaques sur la clef.

On utilise un autre modèle plus fort : celui des attaques par distingueur. Dans ce cas, l'objectif est de tester si les informations recueillies proviennent bien de l'algorithme de chiffrement ciblé ou non. Le résultat de l'attaque est donc un bit : "0" ou "1". Etant donné un système de chiffrement C (respectivement C^*), on définit la probabilité p (respectivement p^*) que l'attaque retourne le résultat "1" lorsque les informations utilisées proviennent effectivement de ce système. On mesure alors la capacité à distinguer C de C^* par la différence $|p - p^*|$ que l'on appelle "avantage". En général, on utilise pour C^* un système idéal de référence. L'intérêt de ce modèle est que si l'on ne

parvient pas à reconnaître C, on ne peut *a fortiori* pas décrypter de message. En effet, si l'on parvient à décrypter avec succès un message, on peut réaliser un distingueur qui répond "`1`" lorsque l'attaque fonctionne avec succès, et "`0`" sinon. Ce modèle d'attaque par distingueur est donc plus fort.[5]

2.2 Cryptanalyse Différentielle

Dans leur ouvrage, Biham et Shamir définissent ainsi la cryptanalyse différentielle.

> "Differential cryptanalysis *is a method which analyzes the effect of particular differences in plaintext pairs on the differences of the resultant ciphertext pairs. These differences can be used to assign probabilities to the possible keys and to locate the most probable key. This method usually works on many pairs of plaintexts with the same particular difference using the resultant ciphertext pairs."*
> Eli Biham and Adi Shamir
> In *Differential Cryptanalysis of the Data Encryption Standard*, 1993.

L'idée fondamentale de cette méthode consiste à exploiter des paires de couples $((x, C_k(x)), (x^*, C_k(x^*)))$ telles que $x^* - x$ et $C_k(x^*) - C_k(x)$ sont des valeurs particulières "intéressantes" afin d'en déduire une information sur la clef k. Donc on note d'ores et déjà deux caractéristiques importantes à cette méthode :

- il faut avoir défini une notion de différence par l'opération "$-$". Cette notion peut être différente en entrée (sur les messages clairs), et en sortie (sur les messages chiffrés).

- Il faut obtenir des paires échantillons $((x, C_k(x)), (x^*, C_k(x^*)))$ avec des différences $x^* - x$ "intéressantes".

Dans la plupart des attaques différentielles, les paires de couples clair-chiffré $((x, C_k(x)), (x^*, C_k(x^*)))$ qui ont une différence $x^* - x$ "intéressante" sont rares. Le modèle d'analyse est donc celui des attaques par texte clair choisi :

[5] La notion de distingueur est originalement due à Turing dans le contexte des générateurs pseudo-aléatoires : on dit qu'un générateur est pseudo-aléatoire s'il n'est pas possible de distinguer ses résultats de ceux d'un générateur aléatoire. (Voir la biographie de Turing [65].)

on suppose que l'attaquant est capable d'obtenir la valeur $C_k(x)$ pour toute valeur x qu'il a choisie.[6]

L'idée originale de la cryptanalyse différentielle est due à Biham et Shamir.[7] Leur objectif était en fait d'attaquer la fonction DES. Il est en fait intéressant de constater qu'une grande partie de la recherche publique a été motivée par un algorithme dont les techniques de constructions sont restées secrètes ![8]

Dans le cadre de ce mémoire, on utilise le cadre formel du chiffrement par blocs en terme de réseau de calcul.

2.2.1 Caractéristique Différentielle Formelle

On définit tout d'abord la notion abstraite de caractéristique d'un réseau de calcul.

Définition 2.1. *Etant donné un réseau de calcul $G = (V, E, \mathrm{Dom})$, on appelle "caractéristique" toute évaluation Ω de G.*

Définition 2.2. *Pour toute fonction f d'un ensemble* $\mathrm{Dom(In)}$ *vers un ensemble* $\mathrm{Dom(Out)}$ *tous deux munis d'une loi interne "$-$", on note*

$$\mathrm{DP}^f(a, b) = \Pr[f(X^*) - f(X) = b / X^* - X = a]$$

pour deux éléments aléatoires X et X^* de $\mathrm{Dom(In)}$ *de distribution uniforme.*

Définition 2.3. *On considère un réseau de calcul interprété que l'on note $G = (V, E, \mathrm{Dom}, f)$ dans lequel chaque ensemble* Dom_a *est muni d'une loi interne "$-$" et muni d'une caractéristique Ω. On note*

$$\mathrm{DP}^G(\Omega) = \prod_{u \in V} \mathrm{DP}^{f_u}(\Omega|_{\mathrm{In}_u}, \Omega|_{\mathrm{Out}_u}).$$

Il est important que $\mathrm{DP}^G(\Omega)$ soit grand, donc en particulier qu'aucune des probabilités DP^{f_u} ne soit nulle. Pour cela, si u est une boîte linéaire, il faut que l'on ait

$$\Omega|_{\mathrm{Out}_u} = f_u\left(\Omega|_{\mathrm{In}_u}\right). \tag{2.1}$$

Cela se traduit ainsi sur des boîtes élémentaires usuelles :

[6] Plus théoriquement, on suppose que l'attaquant dispose d'une source de couples $(X, C_k(X))$ aléatoires pour laquelle il a choisi la distribution de X.

[7] Cette notion est apparue en 1990 lorsque Biham et Shamir ont présenté leur attaque à Crypto 90 [27]. Cet article est publié sous une autre version dans le *Journal of Cryptology* [28]. La première attaque différentielle de DES est parue dans Crypto 92 [29]. Un ouvrage rassemble tous ces résultats dans [30].

[8] Pour un inventaire des attaques de DES, on pourra se référer à une synthèse due à Kusuda et Matsumoto [78].

- pour une boîte linéaire $\oplus$ à deux entrées a_1, a_2 et une sortie b, on a $\Omega_b = \Omega_{a_1} \oplus \Omega_{a_2}$;
- pour une boîte de copie à une entrée a et deux sorties b_1, b_2, on a $\Omega_a = \Omega_{b_1} = \Omega_{b_2}$.

Lorsque ces conditions sont satisfaites, les boîtes linéaires u sont associées à un coefficient DP^{f_u} égal à 1. En revanche, si u est une boîte non linéaire, on distingue deux cas de figure.

- Ou bien $\Omega|_{\mathrm{In}_u}$ est nul, et alors il faut $\Omega|_{\mathrm{Out}_u} = 0$. On a alors $\mathrm{DP}^{f_u} = 1$ et l'on dit que u est "inactive".
- Ou bien $\Omega|_{\mathrm{In}_u}$ est non nul, et l'on dit que u est "active". Le choix de $\Omega|_{\mathrm{In}_u}$ et de $\Omega|_{\mathrm{Out}_u}$ doit alors être judicieux pour que le coefficient $\mathrm{DP}^{f_u}(\Omega|_{\mathrm{In}_u}, \Omega|_{\mathrm{Out}_u})$ ne soit pas trop petit.

2.2.2 Attaque de Biham et Shamir

Etant données une primitive de chiffrement définie par un réseau de calcul interprété $G = (V, E, \mathrm{Dom}, f)$ dans lequel tous les ensembles Dom_a sont munis d'une loi interne "$-$" et une caractéristique Ω, on s'intéresse au chiffrement d'une paire d'entrées aléatoires X et X^*. On s'intéresse à l'événement

$$E_\Omega = [\mathrm{Val}^G(X^*) - \mathrm{Val}^G(X) = \Omega / X^* - X = \Omega|_{\mathrm{In}_G}] \tag{2.2}$$

et à sa probabilité $\Pr[E_\Omega]$ sur la distribution uniforme de X et X^*. En fait, on considère une paire (X, X^*) d'entrées de différence $X^* - X = \Omega|_{\mathrm{In}_G}$, et l'on regarde à chaque étape de calcul $a \in E$ si la paire correspondante $(\mathrm{Val}_a^G(X^*), \mathrm{Val}_a^G(X))$ a pour différence Ω_a. Lorsque ces événements sont simultanément réalisés, on dit que la paire (X, X^*) satisfait la caractéristique. C'est la notion de "paire intéressante".

L'attaque de Biham et Shamir est essentiellement heuristique et suppose un certain nombre d'hypothèses.

Approximation 2.4 (Principe de l'équivalence stochastique). *Avec les notations précédentes, lorsque la probabilité* $\Pr[E_\Omega]$ *est élevée, elle ne dépend pas du choix de la clef et est donc également la probabilité en moyenne sur la distribution de la clef.*[9]

[9]Le principe de l'équivalence stochastique est discuté en particulier dans la thèse de Lai[79, p. 48]. Dans cette étude, on distingue également la notion de "chiffrement de Markov". Par définition, une fonction de chiffrement produit $C = C_r \circ \ldots \circ C_1$ est une

Approximation 2.5. *Avec les notations précédentes, la distribution sur les clefs rend les événements* $\mathrm{Val}_a^G(X^*) - \mathrm{Val}_a^G(X) = \Omega_a$ *indépendants.*

Ces deux approximations entraînent,

$$\Pr[E_\Omega] \approx \mathrm{DP}^G(\Omega). \tag{2.3}$$

Approximation 2.6. *Avec les notations précédentes, lorsque la probabilité* $\mathrm{DP}^G[\Omega]$ *est élevée, on a*

$$\Pr[E_\Omega] \approx \mathrm{DP}^G(\Omega|_{\mathrm{In}_G}, \Omega|_{\mathrm{Out}_G}). \tag{2.4}$$

La caractéristique est dite prédominante.[10]

L'analyse heuristique de Biham et Shamir consiste à déterminer des caractéristiques Ω pour lesquelles $\mathrm{DP}^G(\Omega)$ est élevé.

L'attaque de Biham et Shamir se décompose donc en plusieurs phases.

1. **Phase de saisie.** Obtenir des paires $((X, C_k(X)), (X^*, C_k(X^*)))$ de différence $X - X^*$ choisie.

2. **Phase de filtrage.** Eliminer par une première analyse les mauvaises paires.

3. **Phase d'analyse.** Obtenir par une analyse plus fine des informations sur la clef k.

4. **Phase de recherche.** Effectuer une recherche exhaustive de la clef à l'aide des informations.

On illustre cette méthode par un exemple.

---

fonction de chiffrement de Markov s'il existe une opération de groupe + sur les espaces de blocs telle que pour tout i, Ω_{i-1} et Ω_i, la probabilité

$$\Pr[C_i(x + \Omega_{i-1}) - C_i(x) = \Omega_i]$$

est indépendante de x. (La distribution porte uniquement sur la clef secrète.) Un exemple est réalisé par la construction $C_i(x) = f(x + K_i)$ où K_i est une sous-clef et où f_i est une permutation indépendante de K_i. Pour un tel chiffrement, la seconde hypothèse heuristique de Biham et Shamir est réalisée de manière prouvée.

[10] Il est important de noter que pour certaines attaques différentielles, cette approximation est remise en question, contrairement aux deux autres.

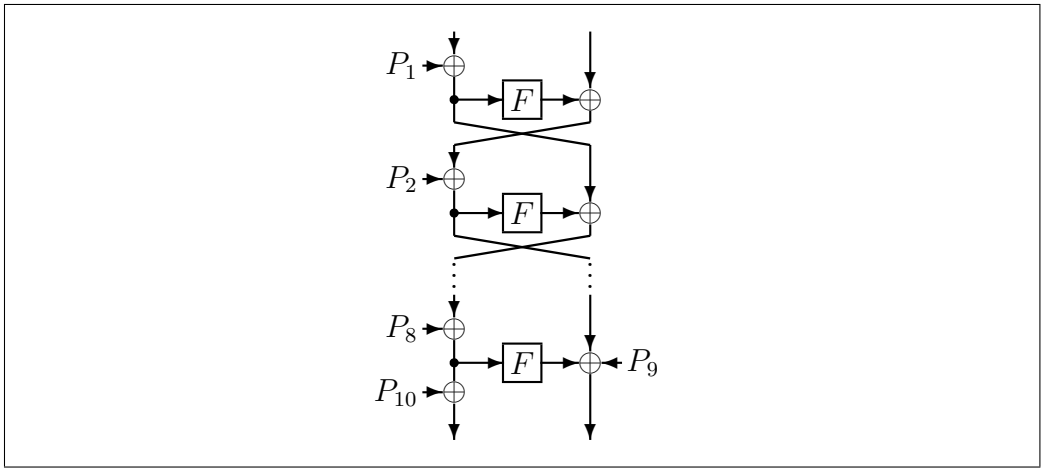


Figure 2.1 – Blowfish Réduit sur Huit Etages.

2.2.3 Attaque Différentielle de Blowfish

Dans cette section, on effectue une attaque différentielle de Blowfish par clefs faibles.[11] Blowfish est un procédé de chiffrement inventé par Bruce Schneier en 1993. C'est un schéma de Feistel à 16 tours sur des blocs de 64 bits. Ce schéma de Feistel offre la particularité de faire apparaître une action d'une sous-clef (voir Fig. 2.1). On se propose ici d'attaquer le schéma réduit à huit tours.

Dans ce schéma, la fonction d'étage F est définie par quatre tables qui représentent des boîtes de calcul qui à toute chaîne de huit bits associe une chaîne de 32 bits. Si les 32 bits d'entrée de F résultent de la concaténation de quatre sous-chaînes de huit bits $a|b|c|d$, on définit

$$F(a|b|c|d) = ((S_1(a) + S_2(b) \bmod 2^{32}) \oplus S_3(c)) + S_4(d) \bmod 2^{32}.$$

L'idée essentielle de Blowfish est que les tables de S_1, S_2, S_3 et S_4 ainsi que les sous-clefs P_i sont engendrées par un générateur pseudo-aléatoire initialisé par la clef secrète.

La présente attaque exploite les collisions possibles sur les boîtes de calcul. Supposons par exemple que la boîte S_1 possède une collision

$$S_1(a) = S_1(a')$$

[11] Blowfish a été présenté au colloque *Fast Software Encryption* en 1993 [136], soit trois ans après l'apparition de la cryptanalyse différentielle. Ce procédé a également été présenté dans le journal du Dr Dobb [137] qui a proposé un concours de la meilleure analyse. La présente section améliore une attaque par clefs faibles présentée au colloque *Fast Software Encryption* en 1996 [164] (voir annexe B) qui a remporté le concours.

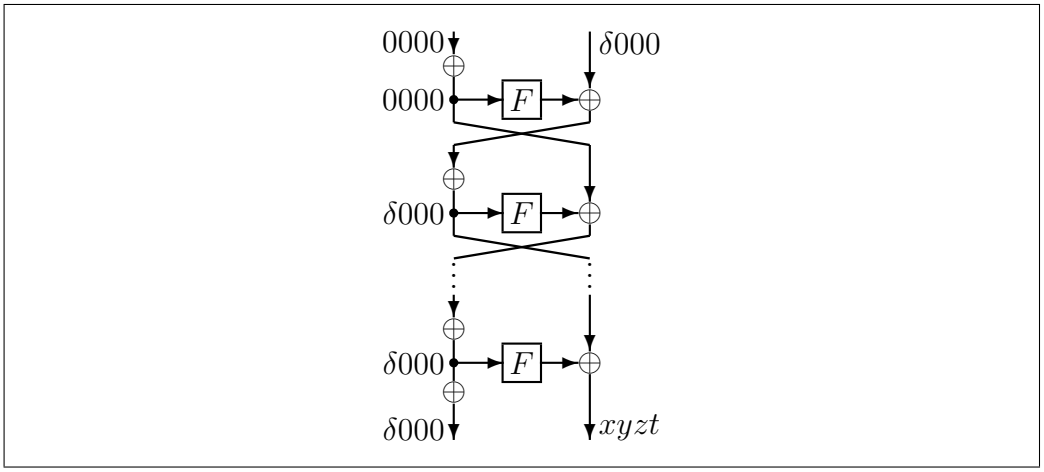


FIGURE 2.2 – Une Caractéristique Différentielle de Probabilité 2^{-21}.

avec $a \neq a'$. Comme S_1 est déterminée aléatoirement à partir de la clef, cela arrive avec une probabilité de

$$1 - \prod_{i=0}^{2^8-1} \left(1 - \frac{i}{2^{32}}\right) = 1 - \frac{2^{32}!}{(2^{32} - 2^8)! \times 2^{32\times 2^8}} \approx 2^{-17.0}. \tag{2.5}$$

On effectue ici une attaque par clefs faibles. Autrement dit, on effectue une attaque préliminaire pour détecter que l'on utilise une clef faible, puis une attaque sur la clef proprement dite.

Lorsque l'on a une collision $S_1(a) = S_1(a')$, si l'on note $\delta = a \oplus a'$, on a

$$\Pr_X[F(X \oplus (\delta, 0, 0, 0)) \oplus F(X) = 0] = 2^{-7}$$

soit, avec les notations,

$$\mathrm{DP}^F((\delta, 0, 0, 0), 0) = 2^{-7}.$$

On considère donc la caractéristique différentielle de Fig. 2.2. Trois étages utilisent la propriété précédente : les étages 2, 4 et 6. Les étages 1, 3, 5 et 7 utilisent une probabilité de 1 car la fonction F est une boîte inactive. Le dernier étage est particulier, car il ne s'intéresse pas à la différence des 32 bits de droite. Suivant l'équation (2.3), la probabilité de la caractéristique est donc 2^{-21}.

L'attaque pour détecter la clef faible fonctionne ainsi.

1. On initialise un compteur c.

2. On effectue exactement $n = 2^{22}$ fois la boucle suivante.

(a) On fait un choix aléatoire de sept chaînes de huit bits B_1, B_2, B_3, B_4, B_6, B_7 et B_8.

(b) (Phase de saisie.) Pour toute chaîne de huit bits B on note

$$\begin{aligned} T_B &= B_1|B_2|B_3|B_4|B|B_6|B_7|B_8 \\ C(T_B) &= C_1^B|C_2^B|C_3^B|C_4^B|C_5^B|C_6^B|C_7^B|C_8^B \\ X_B &= (B \oplus C_1^B)|C_2^B|C_3^B|C_4^B. \end{aligned}$$

(On a donc un jeu de 256 textes clairs choisis.)

(c) (Phase de filtrage.) Si l'on observe une collision $X_B = X_{B'}$, on incrémente c.

3. (Phase d'analyse.) Si l'on a $c > \tau$, on déclare la clef faible. Sinon, on recommence avec une autre clef. (On prend $\tau = 2^8 - 10.2^4 = 96$.)

Le jeu de 256 messages contient exactement 128 paires (B, B') telles que $B \oplus B' = \delta$. Il est facile de voir que si la clef est effectivement faible, chaque jeu contient exactement une paire qui satisfait la caractéristique avec une probabilité $2^{-14.0}$. Dans ce cas, la valeur moyenne de c après n itérations est

$$E(c) = n2^{-14.0}$$

et la variance est

$$V(c) = n2^{-14.0}(1 - 2^{-14.0}) \approx n2^{-14.0}.$$

D'après l'inégalité de Tchebichev[12], on a

$$\Pr[c < \tau] < \frac{V(c)}{(E(c) - \tau)^2}$$

lorsque $\tau < E(c)$, donc

$$\Pr[c < \tau] < \frac{n2^{-14.0}}{(n2^{-14.0} - \tau)^2}.$$

Pour $n = 2^{22}$ et $\tau = 2^8 - 10.2^4$, on a $\Pr[c < \tau] < 1\%$, donc une clef faible sera acceptée avec au moins 99% de chances.

[12] L'inégalité de Tchebichev affirme que pour toute variable aléatoire X, on a

$$\Pr\left[|X - E(X)| > \sqrt{\frac{V(X)}{p}}\right] < p.$$

Des collisions sur X apparaissent "naturellement" avec une probabilité de $2^{-17.0}$ (cela vient en fait de l'équation (2.5)). Dans ce cas, l'espérance de c est

$$E(c) = n2^{-17.0}$$

et la variance est

$$V(c) = n2^{-17.0}(1 - 2^{-17.0}) \approx n2^{-17.0}.$$

D'après l'inégalité de Tchebichev, on a

$$\Pr[c > \tau] < \frac{V(c)}{(\tau - E(c))^2}$$

lorsque $\tau > E(c)$. On a donc

$$\Pr[c > \tau] < \frac{n2^{-17.0}}{(\tau - n2^{-17.0})^2} \approx 2^{-7.0}.$$

La probabilité qu'une clef non faible soit déclarée faible par ce procédé est donc inférieure à 1%. En considérant qu'il faut essayer $2^{17.0}$ clefs, le nombre de "fausses clefs faibles" moyen est donc de 1024 par vraies clefs faibles détectées.

On peut améliorer le nombre de fausses clefs faibles. On peut en effet exploiter la connaissance de $\delta = B \oplus B'$ que l'on obtient dans chaque collision : si les 96 collisions n'indiquent pas la même valeur de δ, c'est que l'on a certainement une fausse clef faible. En fait, pour chaque valeur de δ possible, la probabilité qu'un jeu indique une collision avec cette valeur δ est $2^{-25.0}$, et la probabilité qu'une clef non faible soit déclarée faible en indiquant δ est inférieure à

$$\frac{n2^{-25.0}}{(\tau - n2^{-25.0})^2} \approx 2^{-16.2}$$

et le nombre moyen de fausses clefs faibles pour une vraie clef faible est de 1.8.

L'attaque nécessite 2^{30} textes clairs choisis par clef testée. Il en faut donc environ 2^{47} avant d'obtenir une clef faible. On souligne toutefois que l'inégalité de Tchebichev est souvent très pessimiste, et que l'attaque est certainement bien meilleure. Le travail de l'optimisation de cette attaque particulière est laissée au lecteur (par exemple avec l'aide des bornes de Chernoff).

Etudions maintenant comment exploiter les faiblesses d'une clef.

Une fois la clef faible détectée, on suppose que l'attaquant est capable d'obtenir par une intervention physique la description des tables S_1, S_2, S_3 et

S_4. Cette opération a du sens si l'on considère que la table doit être conservée dans une mémoire assez grande pour contenir 4096 octets. Pour une carte à puce bas-coût, il est raisonnable d'envisager de stocker ces tables à l'extérieur de la carte en ne conservant que les 40 octets $P_1, \ldots, P_{10}$. L'attaquant connaît alors la valeur de δ. De plus, on connaît plusieurs paires $(T_B, T_{B'})$ qui satisfont la caractéristique de Fig. 2.2 d'après la phase de détection de la clef faible. On obtient sur le dernier tour une équation du type

$$F((C_1^B|C_1^B|C_3^B|C_4^B) \oplus P_{10}) \oplus F(((C_1^B \oplus \delta)|C_1^B|C_3^B|C_4^B) \oplus P_{10}) = (x|y|z|t).$$

Seule P_{10} est inconnue ici. On détermine donc cette valeur par une recherche exhaustive. (Il est d'ailleurs possible d'accélérer la recherche en l'isolant sur les octets de P_{10}.) Une attaque semblable permet d'obtenir les autres valeurs $P_1, \ldots, P_9$.

On a donc ici une attaque assez particulière. Elle a une efficacité relative si l'on tient compte du fait

- que c'est une attaque par clefs faibles avec une fraction de $2^{-17.0}$ clefs faibles
- qu'elle nécessite 2^{47} textes clairs choisis avant de détecter une clef faible
- qu'elle nécessite une attaque physique particulière.

Elle met malgré tout en évidence le problème des clefs faibles qui conduisent à des collisions sur des tables S_1, S_2, S_3, S_4 et permet d'illustrer la cryptanalyse différentielle.

2.2.4 Attaques Différentielles Généralisées

Il existe plusieurs types de généralisations de la notion d'attaque différentielle. Tout d'abord, on note que l'approximation de l'équation (2.4) n'est valide que s'il existe une caractéristique prédominante. En effet, avec une différence d'entrée Ω_i et une différence de sortie Ω_o, on a

$$\mathrm{DP}^G(\Omega_i, \Omega_o) = \sum_{\substack{\Omega \\ \Omega|_{\mathrm{In}_G} = \Omega_i, \Omega|_{\mathrm{Out}_G} = \Omega_o}} \Pr[E_\Omega]. \tag{2.6}$$

Il est donc possible de considérer une "somme" de caractéristiques de même Ω_i et Ω_o. Par exemple, Lars Knudsen a imaginé la notion de caractéristique tronquée qui consiste simplement à laisser libres certaines valeurs Ω_a de la caractéristique.

De même, Xuejia Lai a imaginé la notion de caractéristique différentielle d'ordre supérieur. Cette notion immédiate consiste à considérer des paires de paires... Elle présente le défaut de ne pas pouvoir s'itérer aussi bien que dans le cas des caractéristiques différentielles, si bien qu'un faible nombre d'étages rend cette méthode impossible.

2.3 Cryptanalyse Linéaire

La cryptanalyse linéaire utilise des notions duales de la cryptanalyse différentielle. Au lieu d'utiliser la corrélation entre $C_k(x_1)$ et $C_k(x_2)$ connaissant une corrélation choisie entre x_1 et x_2, cette méthode exploite la corrélation linéaire entre x et $C_k(x)$. Plus précisément, étant donnée une forme linéaire $\varphi(x, y)$, on étudie le biais statistique de la distribution de la variable aléatoire $\varphi(X, C_k(X))$. Ceci induit deux différences caractéristiques entre les deux approches :

- les attaques différentielles nécessitent des textes clairs choisis, tandis que seuls des textes clairs connus sont nécessaires pour les attaques linéaires,
- les attaques linéaires extraient une information statistique à partir d'un grand nombre (indivisible) d'échantillons, tandis que les attaques différentielles utilisent un nombre faible de données, mais avec une probabilité de succès faible (ce qui nécessite donc un grand nombre d'itérations).

On a souvent qualifié l'attaque de DES par Matsui (qui nécessite 2^{43} textes clairs connus) meilleure que celle de Biham et Shamir (qui nécessite 2^{47} textes clairs choisis). Il faut cependant relativiser cette comparaison en mentionnant que l'attaque de Matsui utilise un ensemble indivisible de 2^{43} échantillons qui utilisent la même clef, tandis que l'attaque de Biham et Shamir utilise en fait un ensemble de 2^{14} échantillons qui utilisent la même clef et a une probabilité de succès de 2^{-33}. En particulier, si la clef change d'un ensemble à l'autre, cette probabilité est inchangée.

2.3.1 Caractéristique Linéaire Formelle

On définit des notions duales de la cryptanalyse différentielle.

Définition 2.7. *Pour toute fonction f d'un ensemble* Dom(In) *vers un ensemble* Dom(Out) *tous deux munis d'un produit scalaire "$\cdot$" sur* GF(2)*, on note*

$$\mathrm{LP}^f(a, b) = (2\Pr[a \cdot X = b \cdot f(X)] - 1)^2$$

pour un élément aléatoire X de Dom(In) *de distribution uniforme.*

Dans la pratique, on utilise des ensembles $\mathrm{Dom}(A)$ de la forme $\{0,1\}^n$. Par exemple, si l'on a deux chaînes x et y de n bits, $x \cdot y$ désigne la parité du poids de Hamming du "et" bit-à-bit de deux chaînes, soit

$$x \cdot y = \left(\sum_{i=1}^{n} x_i.y_i \right) \bmod 2. \tag{2.7}$$

La quantité $\mathrm{LP}^f(a,b)$ mesure en fait à quel point les bits $a \cdot X$ et $b \cdot f(X)$ sont corrélés et comment on peut approcher l'un en fonction de l'autre. Si ces bits sont égaux avec une probabilité de $\frac{1}{2}$, la quantité LP est nulle, car on ne tire aucun lien entre les deux bits. S'ils sont égaux avec probabilité $\frac{1}{2} + \epsilon$, on a $\mathrm{LP} = 4\epsilon^2$.

Définition 2.8. *On considère un réseau de calcul interprété que l'on note $G = (V, E, \mathrm{Dom}, f)$ dans lequel chaque ensemble* Dom_a *est muni d'un produit scalaire "$\cdot$" sur* GF(2) *et d'une caractéristique Ω. On note*

$$\mathrm{LP}^G(\Omega) = \prod_{u \in V} \mathrm{LP}^{f_u}(\Omega|_{\mathrm{In}_u}, \Omega|_{\mathrm{Out}_u}).$$

On considère une fonction f qui prend en entrée une chaîne de p bits et rend en sortie une chaîne de q bits. Les applications DP^f et LP^f sont liées par une transformation de Fourier. On a en effet

$$\mathrm{LP}^f(a,b) = 2^{-p} \sum_{x,y} (-1)^{(a \cdot x)+(b \cdot y)} \mathrm{DP}^f(x,y) \tag{2.8}$$

et la transformation inverse

$$\mathrm{DP}^f(a,b) = 2^{-q} \sum_{x,y} (-1)^{(a \cdot x)+(b \cdot y)} \mathrm{LP}^f(x,y). \tag{2.9}$$

Il est intéressant de noter les différences dans les règles de construction pour obtenir $\mathrm{DP}^G(\Omega)$ non nul ou $\mathrm{LP}^G(\Omega)$ non nul. En effet, si l'on considère un sommet u du réseau qui représente une application linéaire f_u, pour que les formes linéaires en entrée et en sortie de u soient corrélées, il faut qu'elles soient égales. Pour cela, il faut donc

$${}^t f_u \left(\Omega|_{\mathrm{Out}_u} \right) = \Omega|_{\mathrm{In}_u}. \tag{2.10}$$

Cela se traduit ainsi sur des boîtes élémentaires.

- Pour une boîte linéaire $\oplus$ à deux entrées a_1, a_2 et une sortie b, on a $\Omega_b = \Omega_{a_1} = \Omega_{a_2}$.

- Pour une boîte de copie à une entrée a et deux sorties b_1, b_2, on a $\Omega_a = \Omega_{b_1} \oplus \Omega_{b_2}$.

Ce sont exactement les propriétés inverses de celles qui correspondent aux caractéristiques différentielles.[13]

2.3.2 Attaque de Matsui

Les notions d'attaques linéaires imaginées par Mitsuru Matsui en 1993 sont issues des méthodes d'attaques différentielles et d'idées originales due à Henri Gilbert et Guy Chassé, qui avaient été utilisées contre l'algorithme de chiffrement FEAL-8 en 1990.[14]

Tout comme dans le cas de la cryptanalyse différentielle, on effectue des hypothèses heuristiques.

Approximation 2.9. *Avec les notations précédentes, lorsque les valeurs* $\mathrm{LP}^{f_u}(\Omega|_{\mathrm{In}_u}, \Omega|_{\mathrm{Out}_u})$ *sont grandes, les probabilités correspondantes sont indépendantes.*

Ceci permet de démontrer

$$\mathrm{LP}^{G}(\Omega|_{\mathrm{In}_G}, \Omega|_{\mathrm{Out}_G}) \approx \mathrm{LP}^{G}(\Omega). \tag{2.11}$$

L'équation (2.8) permet de justifier ces hypothèses heuristiques. En effet, si l'on suppose que l'approximation (2.3) est justifiée, d'après les équations (2.6) et (2.8), on a

$$\mathrm{LP}^{G}(\Omega_i, \Omega_o) \approx \sum_{\substack{\Omega \\ \Omega|_{\mathrm{In}_G}=\Omega_i, \Omega|_{\mathrm{Out}_G}=\Omega_o}} \mathrm{LP}^{G}(\Omega). \tag{2.12}$$

Si l'on ne s'intéresse qu'au terme prédominant de la somme, on retrouve donc l'équation (2.11). On établit ainsi que l'approximation 2.9 est la conséquence des approximations 2.4 et 2.5.

Le principe de l'attaque de Matsui consiste à recueillir une information statistique en observant la déviation de la distribution de

$$(\Omega|_{\mathrm{In}_G} \cdot X) \oplus (\Omega|_{\mathrm{Out}_G} \cdot G(X)).$$

[13] Cette dualité des règles de composition des caractéristiques linéaires et différentielles a été remarquée par Biham à Eurocrypt 94 [24].

[14] Matsui a présenté les fondements de la cryptanalyse linéaires à Eurocrypt 93 [89] et Eurocrypt 94 [90]. Ses méthodes ont par la suite été expérimentées contre DES dans des résultats publiés à Crypto 94 [91]. L'article de Gilbert et Chassé de Crypto 90 montre comment casser le système de chiffrement FEAL-8 de la compagnie NTT [54] avec des idées qui ont servi aux fondements de cette technique. La thèse de Gilbert [53] est une bonne référence sur le sujet.

2.3.3 Attaque de Safer

Safer est une fonction de chiffrement développée par James Massey en 1993 qui est décrite dans la section 1.3.2. On remarque

$$\mathrm{LP}^{\mathrm{2PHT}}(\mathtt{00\,01}_x, \mathtt{01\,00}_x) = 1.$$

(On rappelle que les chaînes de bits sont représentées en mode hexadécimal.) En effet, si l'on note x et y les deux entrées de 2PHT, la sortie de gauche est égale à $(2x + y)$ mod 256, et son bit de poids faible est donc égal au bit de poids faible de y. De même, on a

$$\begin{aligned} \mathrm{LP}^{\mathrm{2PHT}}(\mathtt{01\,01}_x, \mathtt{00\,01}_x) &= 1 \\ \mathrm{LP}^{\mathrm{2PHT}}(\mathtt{01\,00}_x, \mathtt{01\,01}_x) &= 1. \end{aligned}$$

Ces observations peuvent conduire à une cryptanalyse linéaire.[15]

Si l'on note PHT la fonction de 64 bits vers 64 bits qui est définie par les trois étages de FFT (voir Fig. 1.7), on a donc

$$\begin{aligned} \mathrm{LP}^{\mathrm{PHT}}(\mathtt{00\,00\,01\,01\,00\,00\,00\,00}_x, \mathtt{00\,00\,01\,01\,00\,00\,00\,00}_x) &= 1 \\ \mathrm{LP}^{\mathrm{PHT}}(\mathtt{00\,01\,00\,00\,00\,01\,00\,00}_x, \mathtt{00\,01\,00\,00\,00\,01\,00\,00}_x) &= 1 \\ \mathrm{LP}^{\mathrm{PHT}}(\mathtt{00\,00\,00\,00\,01\,00\,01\,00}_x, \mathtt{00\,00\,00\,00\,01\,00\,01\,00}_x) &= 1 \\ \mathrm{LP}^{\mathrm{PHT}}(\mathtt{00\,00\,00\,00\,01\,01\,00\,00}_x, \mathtt{00\,00\,01\,00\,00\,00\,01\,00}_x) &= 1 \\ \mathrm{LP}^{\mathrm{PHT}}(\mathtt{00\,01\,00\,01\,00\,00\,00\,00}_x, \mathtt{00\,00\,00\,00\,01\,01\,00\,00}_x) &= 1 \\ \mathrm{LP}^{\mathrm{PHT}}(\mathtt{00\,00\,01\,00\,00\,00\,01\,00}_x, \mathtt{00\,01\,00\,01\,00\,00\,00\,00}_x) &= 1 \end{aligned}$$

On suppose ensuite que l'on a

$$\mathrm{LP}^{P}(\mathtt{01}_x, \mathtt{01}_x) = q$$

ce qui entraîne $\mathrm{LP}^{Q}(\mathtt{01}_x, \mathtt{01}_x) = q$ puisque $Q = P^{-1}$. On remarque qu'une opération $\oplus$ ou une addition modulo 256 avec une constante préserve une totale corrélation entre le bit de poids faible d'entrée et celui de sortie.

Ces remarques permettent de fabriquer six caractéristiques qui utilisent des données calculables à partir de 16 bits de clef et avec un biais LP de q^{10}. Par exemple, si ER représente un étage entier de chiffrement, on a

$$\mathrm{LP}^{\mathrm{ER}}(\mathtt{00\,00\,01\,01\,00\,00\,00\,00}_x, \mathtt{00\,00\,01\,01\,00\,00\,00\,00}_x) = q^2$$

à cause du passage dans les boîtes P et Q (voir Fig. 2.3 sur laquelle on n'a représenté que les Ω_a non nuls de la caractéristique). Si l'on itère cette

[15] Cette attaque de Safer a été présentée à *Fast Software Encryption* 94 [162] (voir annexe D), soit un an après l'apparition de la cryptanalyse linéaire.

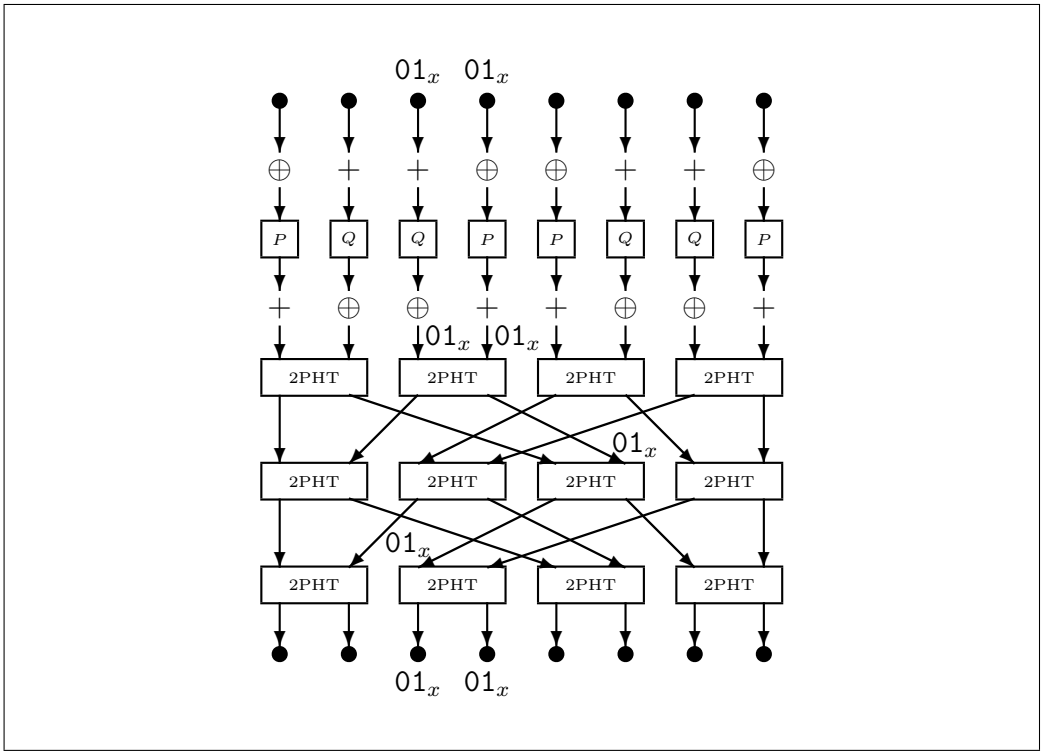


FIGURE 2.3 – Caractéristique sur un Etage de Safer.

caractéristique sur cinq étages, on a donc un biais de q^{10}. On peut prolonger ces cinq étages par la couche "$\oplus$/+" finale (pour laquelle le biais est de un), et remonter jusqu'en sortie des boîtes P et Q du premier étage. Si l'on note x le bloc d'entrée de la fonction de chiffrement, z le bloc de sortie, et si l'on note y le bloc de sortie des boîtes P et Q du premier étage, en posant $z = C(y)$, on a bien

$$\mathrm{LP}^C(00\,00\,01\,01\,00\,00\,00\,00_x, 00\,00\,01\,01\,00\,00\,00\,00_x) = q^{10}$$

(voir Fig. 2.4). En outre, on remarque

$$(y \cdot 00\,00\,01\,01\,00\,00\,00\,00_x) = (Q(x_3 + k_3 \bmod 256) \cdot 01_x) \oplus (P(x_4 \oplus k_4) \cdot 01_x)$$

où x_3 et x_4 représentent respectivement les troisième et quatrième octets de x et où k_3 et k_4 représentent respectivement les troisième et quatrième octets de la première sous-clef. La connaissance de k_3 et k_4 (soit 16 bits de clef) permet donc bien de calculer cette quantité.

Suivant les méthodes de Matsui, on effectue la procédure suivante.

1. **Phase d'analyse.** A toute valeur possible de $K = k_3|k_4$ on associe un compteur c_K. Pour un grand nombre n d'échantillons $(x, \mathrm{Safer}(x))$, on

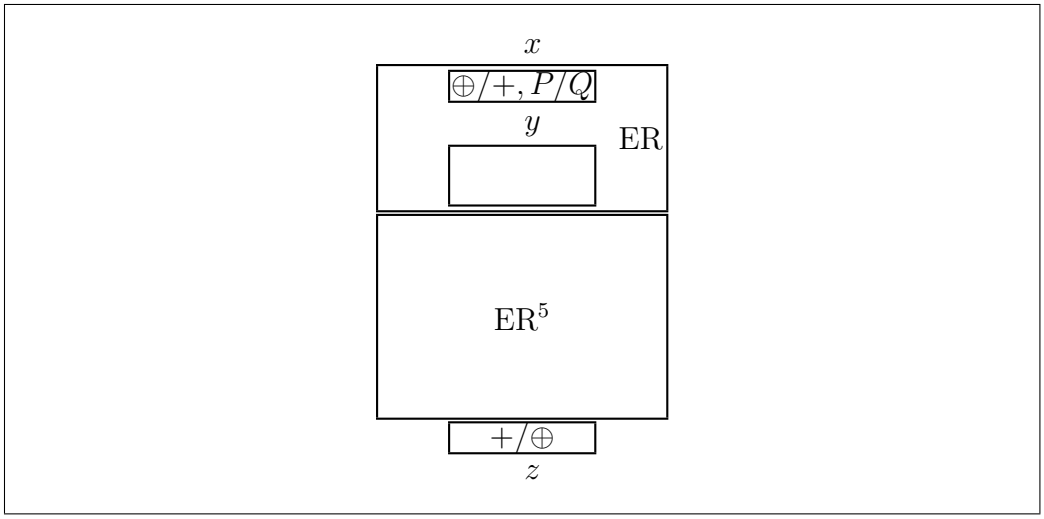


Figure 2.4 – Attaque Linéaire de Safer.

compte le nombre de fois c_K où l'égalité

$$y \cdot 00\,00\,01\,01\,00\,00\,00\,00_x = z \cdot 00\,00\,01\,01\,00\,00\,00\,00_x$$

est réalisée.

2. **Phase de tri.** On trie les valeurs de K dans l'ordre des c_K décroissants ou dans l'ordre inverse.

3. **Phase de recherche.** Pour toute valeur K dans la liste triée, on effectue une recherche exhaustive sur les 56 bits manquants.

En fait, le choix de l'ordre du tri revient à parier sur un bit de la clef. Si l'on se trompe, l'attaque sera mauvaise. Il convient donc d'effectuer les deux choix et de conduire les deux recherches exhaustives en parallèle.

Si l'égalité

$$y \cdot 00\,00\,01\,01\,00\,00\,00\,00_x = z \cdot 00\,00\,01\,01\,00\,00\,00\,00_x$$

admet pour probabilité $\frac{1}{2} + \epsilon_K$, le compteur correspondant suit une distribution qui converge vers une loi normale d'espérance $\frac{n}{2} + n\epsilon_K$ et de variance $n\left(\frac{1}{4} + \epsilon_K^2\right)$, soit

$$\Pr[c_K \leq x] \underset{n\to+\infty}{\longrightarrow} \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\frac{x-\frac{n}{2}-n\epsilon_K}{\sqrt{n\left(\frac{1}{4}-\epsilon_K^2\right)}}} e^{-\frac{t^2}{2}} dt. \tag{2.13}$$

On convient alors de plusieurs approximations.

Approximation 2.10. *La convergence de l'équation (2.13) est très rapide.*

Approximation 2.11. *Dans l'équation (2.13), le terme* $\frac{1}{4} - \epsilon_K^2$ *de la variance peut être approché par* $\frac{1}{4}$.

Approximation 2.12. *Avec les notations précédentes, si* K *est la valeur de la clef réellement utilisée, pour tout* $K' \neq K$, $\epsilon_{K'}$ *est négligeable par rapport à* ϵ_K.

Pour que la recherche exhaustive soit efficace, il faut que la bonne valeur de K corresponde à un compteur c_K éloigné de $\frac{n}{2}$. En supposant que l'on fait un tri dans l'ordre croissant, on peut calculer le rang moyen du bon K dans la liste triée. En effet, le rang moyen est

$$\sum_{K' \neq K} \Pr[c_{K'} < c_K].$$

La variable aléatoire $c_{K'} - c_K$ suit une loi normale d'espérance $n\epsilon_{K'} - n\epsilon_K \approx -n\epsilon_K$ et de variance approchée par $\frac{n}{2}$. Le rang moyen de K est donc environ

$$\frac{2^{16}}{\sqrt{2\pi}} \int_{-\infty}^{\sqrt{2n}\epsilon_K} e^{-\frac{t^2}{2}} dt$$

compte tenu de ces approximations. Si l'on a fait le bon choix pour l'ordre du tri, on a $\epsilon_K < 0$. De plus, on a $4\epsilon_K^2 = q^{10}$. En prenant n tel que $n > 4\lambda^2 q^{-10}$ et $\lambda = 2$, le rang moyen est inférieur à 153, ce qui signifie que l'on récupère ainsi plus de huit bits de la clef.

Telle qu'elle est présentée, l'attaque utilise $16q^{-10}$ échantillons et admet une complexité de $2^{55.3}$ calculs de chiffrement. On peut grandement améliorer la complexité de la recherche en utilisant les autres caractéristiques. Mais cette attaque dépend fortement de $q = \mathrm{LP}^P(\mathtt{01}_x, \mathtt{01}_x)$. Si l'on mesure cette valeur avec la boîte P utilisée dans Safer, on obtient $q = 0$, si bien que l'attaque ne peut fonctionner ! On peut cependant s'interroger sur l'origine de ce $q = 0$. On démontre facilement que si l'on prend une permutation aléatoire de l'ensemble des octets pour définir P, la probabilité d'obtenir $q = 0$ est de 10%. On peut se demander si le créateur de l'algorithme Safer a délibérément choisi P pour obtenir $q = 0$ ou s'il a eu cette chance de 10%. On relativise toutefois cette remarque en observant

1. que l'attaque précédente n'a de sens que si elle est praticable pour un nombre d'échantillons inférieur à 2^{64}, soit pour $q \geq 2^{-6}$, ce qui est vrai avec une probabilité de 6% seulement, donc le créateur de Safer aurait dû être malchanceux pour que cette attaque fonctionne

2. que $q = 0$ est une propriété "naturelle" de toute fonction exponentielle. En effet, si g est un générateur de $\mathbf{Z}_{257}^*$, on a

$$g^{128} \equiv -1 \pmod{257}$$

donc

$$g^{x+128} \bmod 257 = 257 - (g^x \bmod 257).$$

Donc, $P(x \oplus \texttt{80}_x)$ et $P(x)$ ont nécessairement un bit de poids faible différent. Il ne peut donc y avoir aucune corrélation entre les bits de poids faibles de x et de $P(x)$.

Cette "attaque manquée" montre cependant qu'il est dangereux d'utiliser des boîtes 2PHT qui laissent ainsi passer un bit sans biais.

2.3.4 Attaques Linéaires Généralisées

Des tentatives de généralisation de la cryptanalyse linéaire ont été menées. Notamment, l'attaque de Matsui utilisant des approximations du type

$$f(\text{entrée}) \oplus g(\text{sortie}) \approx h(\text{clef})$$

où f, g et h sont linéaires. Harpes *et al.* ont proposé la notion de "somme de trois termes" non nécessairement linéaire et ont effectué la même analyse. Cette idée n'a cependant pas encore débouché sur de réelles applications.[16]

Une autre idée consiste à chercher à utiliser simultanément plusieurs caractéristiques. Une analyse heuristique montre alors que si l'on utilise de telles caractéristiques sur les mêmes informations, cela revient à remplacer le biais LP d'une caractéristique unique par la somme du biais de toutes les caractéristiques.[17] Par exemple, l'attaque de Matsui contre DES utilise deux caractéristiques de même biais, ce qui revient à améliorer le nombre d'échantillons nécessaires d'un facteur 2.

Dans un autre type de généralisation, on occulte complètement l'aspect heuristique de la construction de caractéristiques linéaires pour ne retenir que la méthode statistique. Dans cette attaque, on utilise des fonctions h_1, h_2 et h_3 (dont le rôle est d'extraire l'information utile) et l'on étudie la distribution de

$$Y = h_3(K, h_2(X, C_k(X)))$$

pour des fonctions d'extraction d'information h_1, h_2 et h_3 et pour un X aléatoire. Lorsque l'on a $K = h_1(k)$, la distribution doit être biaisée, et

[16] La notion de somme de trois termes est parue dans [60].

[17] Cette analyse est due à Kaliski et Robshaw [71].

lorsque $K \neq h_1(k)$, elle doit être une distribution standard (la distribution uniforme par exemple). L'attaque utilise une statistique que l'on modélise par une fonction $\Sigma(y_1, \ldots, y_n)$ symétrique : pour toute permutation σ de $\{1, \ldots, n\}$, on a

$$\Sigma(y_{\sigma(1)}, \ldots, y_{\sigma(n)}) = \Sigma(y_1, \ldots, y_n).$$

L'attaque se décompose en quatre étapes.

1. **Phase de comptage.** On récupère n échantillons $s_i = h_2(x_i, C_k(x_i))$ en comptant leurs occurrences dans une table. On a ainsi un compteur n_j égal au nombre de valeurs i pour lesquelles $s_i = j$.

2. **Phase d'analyse.** Pour chaque valeur K, on calcule $y_i = h_3(K, s_i)$ et l'on évalue une statistique $\Sigma(y_1, \ldots, y_n)$ pour affecter une note M_K à la valeur K. (Comme Σ est symétrique, on utilise en fait les nombres d'occurrences de $y_i = h_3(K, s_i)$ ce qui garantit un nombre de calcul indépendant de n.)

3. **Phase de tri.** On trie les valeurs K possibles dans l'ordre des notes M_K les plus vraisemblables.

4. **Phase de recherche.** Pour chaque valeur K dans la liste triée, on essaie toutes les clefs possibles k' telles que $h_1(k') = K$ jusqu'à avoir $k' = k$.

La cryptanalyse linéaire de Safer est un exemple particulier de ce type d'attaque. Avec les notations de la section 2.3.3, la fonction h_1 extrait de la clef secrète les 16 bits $k_3|k_4$; la fonction h_2 extrait de (x, z) les 16 octets de x nécessaires au calcul de y_3 et y_4 et le bit

$$z \cdot 00\,00\,01\,01\,00\,00\,00\,00_x$$

et la fonction h_3 calcule le bit

$$(y \cdot 00\,00\,01\,01\,00\,00\,00\,00_x) \oplus (z \cdot 00\,00\,01\,01\,00\,00\,00\,00_x).$$

La phase de comptage utilise donc 2^{17} compteurs. La phase d'analyse effectue 2^{16} itérations pour calculer un compteur c_K (le nombre de valeurs $h_3(K, s)$ égales à 1) et calcule une note $M_K = |c_K - \frac{n}{2}|$.

Afin d'analyser la complexité de cette attaque, on effectue les approximations suivantes.

Approximation 2.13. *Lorsque $K \neq h_1(k)$, la distribution de*

$$h_3(K, h_2(X, C_k(X)))$$

est une distribution D indépendante de K.

Approximation 2.14. *Lorsque $K = h_1(k)$, la distribution de*

$$h_3(K, h_2(X, C_k(X)))$$

est une distribution D' indépendante de D.

Approximation 2.15. *Lorsque $Y_1, \ldots, Y_n$ suivent indépendamment la distribution D (respectivement D'), la statistique $\Sigma(Y_1, \ldots, Y_n)$ suit une distribution normale d'espérance μ et d'écart-type σ (respectivement μ' et σ').*

Approximation 2.16. *On a $\sigma \approx \sigma'$.*

On note

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}} dt.$$

On note κ le nombre de clefs possibles, ℓ le nombre de $h_1(k)$ possibles (on suppose en outre que h_1 est équilibrée : si k admet une distribution uniforme, alors $h_1(k)$ également), n le nombre d'échantillons, et s le nombre de $h_2(x, C_k(x))$ possibles. La complexité des phases de comptage, d'analyse et de tri est n, s et $\ell \log \ell$ respectivement. On démontre que la complexité de la recherche exhaustive est de

$$\frac{\kappa}{\ell} + \left(\kappa - \frac{\kappa}{\ell}\right) \Phi\left(-\frac{\mu' - \mu}{\sigma\sqrt{2}}\right).$$

Dans le cas de l'attaque de Safer, on a $\ell = 2^{16}$, $s = 2^{17}$ et

$$\left(\frac{\mu' - \mu}{\sigma}\right)^2 = q^{10}.$$

(C'est le coefficient LP de la caractéristique.)

Le problème, pour mettre en œuvre cette généralisation de l'attaque de Matsui, est qu'il faut trouver un autre moyen heuristique pour obtenir un bon choix de h_1, h_2, h_3 et Σ. Il est possible d'obtenir h_1, h_2 et h_3 par des méthodes de projection présentées dans la section 2.4.6. Un bon Σ lorsque l'on ne sait pas *a priori* comment récupérer une information statistique cachée est le test du χ^2. Cela permet d'effectuer une attaque contre la fonction de chiffrement DES légèrement meilleure que l'attaque de Matsui : on obtient une attaque de complexité $2^{42.9}$ contre $2^{43.0}$... (Cela représente tout de même un gain de 7% !) Ces idées sont reportées en annexe C.[18]

[18] Elles ont été publiées à la conférence ACM CCS 96 [165].

2.4 Attaques Génériques

La notion de réseau de calcul définie dans la section 1.1.6 distingue la notion géométrique de graphe de celle d'interprétation. Avant d'étudier les propriétés d'une interprétation, il paraît naturel de se demander si le graphe lui-même possède des faiblesses. Pour cela, on définit la notion de graphe d'équations, et l'on considère les méthodes de résolution formelles par des algorithmes dits "génériques". Ceux-ci généralisent en fait assez naturellement les notions d'attaques par "recherche exhaustive" et d'"attaque dans le milieu".[19]

2.4.1 Modèle Général de Graphe d'Equations

Par définition, un réseau de calcul est un graphe orienté sans cycle dans lequel chaque arête est étiquetée par un ensemble dont le cardinal définit le "degré de liberté". Les sommets d'entrée correspondent à des variables ou à des paramètres, et les sommets de sortie à des résultats. Par exemple, pour une fonction de chiffrement, on a un texte clair et une clef en entrée, et un texte chiffré en sortie. Si l'on s'intéresse à la recherche d'une clef en connaissant un texte clair et un texte chiffré, cela revient à restreindre le degré de liberté de certaines arêtes. Dans un tel problème, la notion d'orientation du graphe intervient peu, car on cherche en général à remonter les calculs en sens inverse. On définit ainsi la notion de "graphe d'équations".

On rappelle tout d'abord quelques définitions et notations.

Définition 2.17. *On appelle graphe non orienté la donnée (V, E) d'un couple formé d'un ensemble V et d'un ensemble E de paires $\{u, v\}$ d'éléments u et v de V. (On impose $u \neq v$: le graphe est sans boucle.) V est l'ensemble des "sommets", et E est l'ensemble des "arêtes". Une arête $a = \{u, v\}$ est "adjacente" à u et v. Pour un sommet u, on note Adj_u l'ensemble des arêtes adjacentes à u. On notera également $a = uv$ par commodité.*

Définition 2.18. *Un "graphe d'équations" est la donnée*

$$G = (V, E, \mathrm{Dom}, \mathrm{Sol})$$

d'un graphe non orienté (V, E), d'un étiquetage Dom *qui à chaque arête a associe un ensemble Dom_a et d'un étiquetage* Sol *qui à chaque sommet u associe un réel positif Sol_u inférieur au cardinal de $\mathrm{Dom}(\mathrm{Adj}_u)$.*

[19] Ces notions d'attaques génériques ont pour la première fois été présentées à la conférence *Fast Software Encryption* 93 [145] pour proposer une fonction de hachage. Elles ont été reprises à Eurocrypt 94 [146], puis publiées dans le *Journal of Cryptology* [147] (voir annexe E).

Intuitivement, chaque arête est une inconnue, et chaque sommet est une équation locale entre les inconnues adjacentes. L'entier $\#\mathrm{Dom}_a$ est la taille du domaine de l'inconnue a, et Sol_u est la taille de l'ensemble des solutions.

Comme on s'intéresse à la difficulté intrinsèque de résoudre l'équation indépendamment de la description des équations locales, on considère chaque équation locale comme une équation aléatoire.

Définition 2.19. *Un "graphe d'équations aléatoire interprété" est la donnée* $G = (V, E, \mathrm{Dom}, \mathrm{Sol}, F)$ *d'un graphe d'équations* $G = (V, E, \mathrm{Dom}, \mathrm{Sol})$ *et d'une "interprétation aléatoire"* F*, qui est une application aléatoire de distribution donnée qui à tout sommet* u *associe un sous-ensemble noté* F_u *de* $\mathrm{Dom}(\mathrm{Adj}_u)$ *tel que* $E(\#F_u) = \mathrm{Sol}_u$*.*

En fait, une équation locale u est définie par son ensemble de solutions F_u. Une solution est formalisée par une application qui à toute arête adjacente a associe une valeur de Dom_a : c'est une évaluation partielle de G définie sur Adj_u. Une interprétation aléatoire F est donc définie par un ensemble aléatoire $F_u \subseteq \mathrm{Dom}(\mathrm{Adj}_u)$. On dit qu'une partie de $\mathrm{Dom}(A)$ est un ensemble de solutions partielles définies sur A. On introduit également une nouvelle notation.

Définition 2.20. *Soit un graphe d'équations* $G = (V, E, \mathrm{Dom}, \mathrm{Sol})$*. Si l'on a* $X \subseteq \mathrm{Dom}(A)$ *et* $Y \subseteq \mathrm{Dom}(B)$*, on définit la "jointure" de* X *et* Y

$$X \bowtie Y = \{t \in \mathrm{Dom}(A \cup B); t|_A \in X, t|_B \in Y\}.$$

Si X et Y sont respectivement des ensembles de solutions partielles définies sur des domaines A et B, $X \bowtie Y$ est l'ensemble des solutions partielles sur $A \cup B$.

2.4.2 Résolution de Graphe d'Equations

Un algorithme de résolution générique procède par étapes successives en gérant des ensembles d'évaluations partielles définies sur des ensembles d'arêtes de plus en plus grands. Comme il traite les solutions de manière générique, il ne peut qu'"ouvrir" successivement l'ensemble des solutions d'une équation locale et en sélectionner aléatoirement dans cet ensemble sans ségrégation possible. On lui autorise donc les deux types d'opérations élémentaires suivantes.

- La restriction aléatoire d'un ensemble de solutions partielles X pour en obtenir une fraction p en moyenne. On note cette opération $\gamma_p(X)$. (Formellement, on a $\gamma_p(X) \subseteq X$ et pour tout $t \in X$, les événements $t \in \gamma_p(X)$ sont indépendants et de probabilité p.)

- La jointure de deux ensembles de solutions partielles X et Y que l'on note $X \bowtie Y$.

Plus précisément, un algorithme est une suite finie $X_1, X_2, \ldots, X_n$ d'ensembles d'évaluations partielles telle que pour tout i, on a

- soit $X_i = F_u$ pour un sommet u ;
- soit $X_i = X_j \bowtie X_k$ pour des valeurs $j < i$ et $k < i$;
- soit $X_i = \gamma_p(X_j)$ pour une valeur $j < i$ et un réel p.

Pour que la résolution soit correcte, il faut que l'on ait

$$X_n \subseteq \bowtie_{u \in V} F_u.$$

Suivant le contexte, on pourra considérer que le but de l'algorithme est d'obtenir l'ensemble de toutes les solutions, ou une seule solution. Dans la suite, on adopte une définition "visuelle" d'un algorithme de résolution au moyen de la notion formelle de terme.

Définition 2.21. *Etant donné un graphe d'équations $G = (V, E, \mathrm{Dom}, \mathrm{Sol})$, un "algorithme de résolution" est défini par un terme R composé exclusivement*

- *d'opérateurs d'arité 0 notés σ_u (pour tout sommet u de G),*
- *d'opérateurs d'arité 1 notés γ_p (pour tout réel p de $[0, 1]$),*
- *d'opérateurs d'arité 2 notés $\bowtie$,*

et tel que chaque sommet u de G est représenté par au moins un opérateur σ_u. Etant donnée une interprétation aléatoire f de G, tout sous-terme S de R définit un ensemble aléatoire de solutions partielles $\mathrm{Val}_f(R)$ par induction suivant les règles suivantes :

- $\mathrm{Val}_f(\sigma_u) = f_u$;
- $\mathrm{Val}_f(\gamma_p(S)) = \gamma_p(\mathrm{Val}_f(S))$;
- $\mathrm{Val}_f(S \bowtie T) = \mathrm{Val}_f(S) \bowtie \mathrm{Val}_f(T)$.

Avec une telle définition, on montre que $\mathrm{Val}_f(R)$ est nécessairement un ensemble de solutions du graphe d'équations.

2.4.3 Complexité d'un Algorithme de Résolution

On formalise ainsi la complexité d'un algorithme de résolution.

Définition 2.22. *Etant donné un graphe d'équations $G = (V, E, \mathrm{Dom}, \mathrm{Sol})$ muni d'une interprétation f et d'un algorithme de résolution R, la "complexité" $\mathrm{Comp}_f(R)$ de R est le maximum de $\#\mathrm{Val}_f(S)$, pour tout sous-terme S de R.*

Exemple 2.23 (Calcul direct). On suppose que l'on s'intéresse au chiffrement $C_k(x)$ d'une donnée x au moyen d'une clef k. Lorsque l'algorithme de chiffrement C provient d'un réseau de calcul interprété $G = (V, E, \mathrm{Dom}, f)$, on a un graphe orienté sans cycles (V, E) qui permet de calculer $C_k(x)$ au moyen de k et x. Comme le graphe est sans cycles, on peut énumérer tous les sommets de G dans un ordre $u_1, \ldots, u_n$ tel que pour tout i et tout sommet v, si (v, u_i) est une arête de G, alors $v = u_j$ pour un certain j tel que $j < i$. En particulier, u_1 est nécessairement un sommet d'entrée de G, et u_n est nécessairement un sommet de sortie.

Le graphe (V, E) définit naturellement un graphe non orienté (V, E') avec le même ensemble de sommets. Pour tout sommet u, on définit Sol_u par

- $\mathrm{Sol}_u = 1$ si u est un sommet d'entrée ;
- $\mathrm{Sol}_u = \#\mathrm{Dom}(\mathrm{In}_u)$ sinon.

Lorsque l'on connaît x et k, il n'y a effectivement qu'une "solution" pour les sommets d'entrée. On a ainsi un graphe d'équations $G' = (V, E, \mathrm{Dom}, \mathrm{Sol})$ pour lequel on peut définir un algorithme de résolution

$$R = (\ldots((\sigma_{u_1} \bowtie \sigma_{u_2}) \bowtie \sigma_{u_3}) \bowtie \ldots) \bowtie \sigma_{u_n}.$$

Cet algorithme trivial consiste à calculer successivement les boîtes de calcul dans un ordre tel que leurs entrées sont déjà résolues.

On définit une interprétation f' de $G' = (V, E', \mathrm{Dom}, \mathrm{Sol})$ en complétant f sur In_G et Out_G. Pour un sommet d'entrée u, on définit f'_u par un singleton qui représente la valeur de u, et pour un sommet de sortie v, on définit $f'_v = \mathrm{Dom}(\mathrm{Adj}_v)$. Pour l'interprétation f', on montre que $\mathrm{Comp}_{f'}(R)$ est

$$\mathrm{Comp}_{f'}(R) = \max_x \#\mathrm{Dom}(\mathrm{In}_u)$$

qui est en général faible. (Ces algorithmes génériques ont en fait une forme de "stupidité" qui les borne à évaluer une fonction f_u sur une entrée t en énumérant toutes les possibilités de $\mathrm{Dom}(\mathrm{In}_u)$ jusqu'à trouver t.)

Exemple 2.24 (Recherche exhaustive). On suppose que l'on s'intéresse à la recherche d'une clef k pour une fonction de chiffrement C_k avec la connaissance d'un couple $(x, C_k(x))$. On utilise les notations de l'exemple précédent en modifiant la définition de Sol. On suppose que la clef k intervient par un unique sommet d'entrée v et que tous les sommets de sortie correspondent à $C_k(x)$. On définit

- $\mathrm{Sol}_u = 1$ si u est un sommet d'entrée différent de k ;
- $\mathrm{Sol}_u = K$ si $u = v$ (K étant le nombre de clefs possibles) ;
- $\mathrm{Sol}_u = 1$ si u est un sommet de sortie ;
- $\mathrm{Sol}_u = \#\mathrm{Dom}(\mathrm{In}_u)$ sinon.

Le même algorithme commence par calculer directement ce qu'il peut, soit les sommets $u_1, \ldots, u_r$ en n'ayant à gérer que des ensembles de solutions partielles réduites à un seul élément, puis tombe sur $u_{r+1} = v$. On note

$$S \bowtie \sigma_v$$

le sous-terme correspondant. Au moment d'évaluer $\mathrm{Val}_f(S \bowtie \sigma_v)$, il faut calculer f_v qui contient toutes les clefs possibles. L'algorithme progressera ensuite en gérant des ensembles de K solutions partielles qui correspondent à toutes les clefs possibles. Au moment où l'algorithme tombera sur un sommet de sortie, la valeur de $C_k(x)$ étant imposée, des solutions seront éliminées, et l'on ne retiendra à la fin que les clefs k telles que $C_k(x)$ est bien la valeur attendue.

Lorsque $C_k(x)$ est une chaîne de n bits, si $K > 2^n$, il est judicieux de prendre $\gamma_{2^n/K}(\sigma_v)$ à la place de σ_v. On obtiendra ainsi en moyenne une seule clef, si l'interprétation a une distribution suffisamment régulière.

Afin d'évaluer la complexité des algorithmes, on suppose que la distribution de l'interprétation admet les propriétés suivantes qui respectent la "généricité" des algorithmes de résolution.

Définition 2.25. *Pour un graphe d'équations* $(V, E, \mathrm{Dom}, \mathrm{Sol})$*, une partie aléatoire* X *de* $\mathrm{Dom}(A)$ *est dite "localement uniforme" si les solutions potentielles* $t \in \mathrm{Dom}(A)$ *sont équiprobables dans* X*.*

Une interprétation aléatoire F *est dite "localement uniforme" si* F_u *est localement uniforme pour tout sommet* u*. On a donc*

$$\Pr[t \in F_u] = \frac{\mathrm{Sol}_u}{\prod\limits_{a \in \mathrm{Adj}_u} \#\mathrm{Dom}_a}.$$

De nombreuses distributions "raisonnables" sur les parties de $\mathrm{Dom}(\mathrm{Adj}_u)$ vérifient cette hypothèse :

1. la distribution uniforme sur les parties de $\mathrm{Dom}(\mathrm{Adj}_u)$ de cardinal égal à Sol_u ;
2. la distribution

$$\{t \in \mathrm{Dom}(\mathrm{Adj}_u); t|_{\mathrm{Out}_u} = T(t_{\mathrm{In}_u})\}$$

induite par une fonction aléatoire T de $\mathrm{Dom}(\mathrm{In}_u)$ vers $\mathrm{Dom}(\mathrm{Out}_u)$ et de distribution uniforme, lorsqu'il existe une partition $\mathrm{In}_u \cup \mathrm{Out}_u = \mathrm{Adj}_u$ telle que $\prod_{a \in \mathrm{In}_u} \#\mathrm{Dom}_a = \mathrm{Sol}_u$.

Définition 2.26. *Une interprétation aléatoire F est dite homogène si les F_u sont indépendants pour tous les sommets u.*

Dans la suite, on ne considérera que des interprétations localement uniformes et homogènes. Ces propriétés permettent de calculer la complexité des algorithmes indépendamment de l'interprétation.

On note que l'on peut supprimer les opérations γ_p des algorithmes en diminuant éventuellement Sol sans affecter le nombre moyen de solutions que l'on obtient en fin de résolution ni leur complexité. Pour cela, il suffit d'observer que $\gamma_p(X \bowtie Y)$ a la même distribution que $\gamma_q(X) \bowtie \gamma_r(Y)$ lorsque $qr = p$. Cela permet de faire descendre les opérations γ_p jusque vers les feuilles du terme. Si un terme utilise ensuite plusieurs occurrences d'un sommet u avec des opérations différentes $\gamma_{p_1}(\sigma_u), \ldots, \gamma_{p_n}(\sigma_u)$, comme la résolution devra à un moment effectuer une intersection de ces sélections, on peut remplacer F_u par $\gamma_{p_1 \ldots p_n}(F_u)$ dans la résolution en supprimant les opérations γ. Cela revient à remplacer Sol_u par $p_1 \ldots p_n \mathrm{Sol}_u$. Dans la suite, on ne considère donc que des algorithmes sans fonction γ_p.

2.4.4 Complexité Théorique

On a le lemme suivant.

Lemme 2.27. *Soit un graphe d'équations $G = (V, E, \mathrm{Dom}, \mathrm{Sol})$ muni d'une interprétation aléatoire F homogène et localement uniforme. Pour tout algorithme de résolution R sans fonctions γ_p, pour tout sous-terme S de R on a*

$$E(\#\mathrm{Val}_F(S)) = \prod_{u \in V(S)} \mathrm{Sol}_u \prod_{a \in E|_{V(S)}} (\#\mathrm{Dom}_a)^{-1}$$

où $V(S)$ est l'ensemble des sommets u de G représentés par au moins un opérateur σ_u dans S, et $E|_{V(S)}$ est l'ensemble des arêtes de G dont les deux extrémités sont dans $V(S)$.

Preuve. Comme S n'a pas d'opérateur γ_p, on a

$$\mathrm{Val}_F(S) = \bowtie_{u \in V(S)} F(u).$$

On a

$$\begin{aligned} E(\#\mathrm{Val}_F(S)) &= \sum_{t \in \mathrm{Dom}(\mathrm{Adj}(V(S)))} \Pr[t \in \mathrm{Val}_F(S)] \\ &= \sum_{t \in \mathrm{Dom}(\mathrm{Adj}(V(S)))} \prod_{u \in V(S)} \Pr[t|_{\mathrm{Adj}(u)} \in F_u] \\ &= \#\mathrm{Dom}(\mathrm{Adj}(V(S))) \times \prod_{u \in V(S)} \frac{\mathrm{Sol}_u}{\prod_{a \in \mathrm{Adj}(u)} \#\mathrm{Dom}_a} \\ &= \prod_{a \in \mathrm{Adj}(V(S))} \#\mathrm{Dom}_a \times \prod_{u \in V(S)} \frac{\mathrm{Sol}_u}{\prod_{a \in \mathrm{Adj}_u} \#\mathrm{Dom}_a} \end{aligned}$$

où $\mathrm{Adj}(V(S))$ est l'ensemble des arêtes adjacentes à au moins un sommet de $V(S)$. Les arêtes dont les deux extrémités sont dans $V(S)$ sont comptées deux fois dans le second produit. Celles dont une seule extrémité est dans $V(S)$ ne sont comptées qu'une seule fois. On obtient donc ainsi l'équation annoncée. □

Si l'on écrit l'équation du lemme 2.27 de manière logarithmique, on obtient naturellement la notion de forme quadratique suivante.

Définition 2.28. *A tout graphe d'équations $G = (V, E, \mathrm{Dom}, \mathrm{Sol})$ on associe une matrice Γ dont les lignes et les colonnes sont définies par un sommet de G. Pour tout couple de sommets (u, v) on définit le coefficient $\Gamma_{u,v}$ par*

$$\Gamma_{u,v} = \begin{cases} -\frac{1}{2} \log \#\mathrm{Dom}_{\{u,v\}} & \text{si } \{u, v\} \in E \\ \log \mathrm{Sol}_u & \text{si } u = v \\ 0 & \text{sinon.} \end{cases}$$

Pour toute fonction g de V vers $\mathbf{R}$, on définit

$$\Gamma(g) = \sum_{u,v \in V} \Gamma_{u,v} g(u) g(v).$$

On dit que Γ est la "forme quadratique associée à G". Par convention, pour tout ensemble de sommets U, on définit

$$\log \mathrm{Sol}(U) = \Gamma(U) = \sum_{u,v \in U} \Gamma_{u,v}.$$

Le lemme 2.27 s'écrit donc $E(\#\mathrm{Val}_F(S)) = \mathrm{Sol}(V(S))$. On obtient ainsi le théorème suivant.

Théorème 2.29. *Soit un graphe d'équations $G = (V, E, \mathrm{Dom}, \mathrm{Sol})$ muni d'une interprétation aléatoire F homogène et localement uniforme. Pour tout algorithme de résolution R sans fonctions γ_p, si l'on note $\#R$ la taille de R (le nombre de ses sous-termes), on a*

$$\max_S \mathrm{Sol}(V(S)) \leq E_F(\mathrm{Comp}_F(R)) \leq \#R \times \max_S \mathrm{Sol}(V(S))$$

où S est un sous-terme quelconque de R.

Preuve. Comme l'espérance des maxima est supérieure au maximum des espérances, l'inégalité de gauche est une conséquence directe du lemme 2.27. Pour montrer l'inégalité de droite, on a

$$\begin{aligned} E_F(\mathrm{Comp}_F(R)) &= E\left(\max_{S \text{ sous-terme de } R} \#\mathrm{Val}_F(S)\right) \\ &\leq E\left(\sum_{S \text{ sous-terme de } R} \#\mathrm{Val}_F(S)\right) \\ &= \sum_{S \text{ sous-terme de } R} E(\#\mathrm{Val}_F(S)) \\ &= \sum_{S \text{ sous-terme de } R} \mathrm{Sol}(V(S)) \\ &\leq \#R \times \max_{S \text{ sous-terme de } R} \mathrm{Sol}(V(S)). \end{aligned}$$

□

On définit donc un estimateur de la complexité moyenne d'un algorithme.

Définition 2.30. *Pour un algorithme de résolution R sans opérateur γ_p d'un graphe d'équations $G = (V, E, \mathrm{Dom}, \mathrm{Sol})$, on définit la "complexité théorique"*

$$\mathrm{Comp}'(R) = \max_{S \text{ sous-terme de } R} \mathrm{Sol}(V(S)).$$

On a le corollaire suivant.

Corollaire 2.31. *On considère un graphe d'équations muni d'une interprétation aléatoire F localement uniforme et homogène. Tout algorithme de résolution de taille t vérifie les inégalités*

$$\mathrm{Comp}'(R) \leq E_F(\mathrm{Comp}_F(R)) \leq t.\mathrm{Comp}'(R).$$

Ce résultat montre que l'on peut étudier les faiblesses induites par la structure de graphe seule. En effet, si l'on suppose que les équations locales ont les propriétés d'uniformité et d'homogénéité voulues, la complexité théorique d'un algorithme de résolution ne dépend plus de leur distribution.

Exemple 2.32 (Recherche exhaustive). On reprend l'étude de l'exemple 2.24 pour résoudre l'équation $C_k(x) = y$ à partir de x et y. Comme on ne s'intéresse pas à la définition précise de l'algorithme de chiffrement, on peut en fait simplifier le problème au moyen du graphe de la figure 2.5 :

- on a un ensemble de sommets $V = \{u_x, u_k, u_y, u_C\}$;
- on a un ensemble d'arêtes

$$E = \{u_x u_C, u_k u_C, u_y u_C\};$$

- on a $\mathrm{Dom}_{u_x u_C} = \mathrm{Dom}_{u_k u_C} = M$, $\mathrm{Dom}_{u_y u_C} = K$;
- on a $\mathrm{Sol}_{u_x} = \mathrm{Sol}_{u_y} = 1$, $\mathrm{Sol}_{u_k} = \#K$, $\mathrm{Sol}_{u_C} = \#M \times \#K$;
- on a $F_{u_x} = \{x\}$, $F_{u_y} = \{y\}$ et $F_{u_k} = K$.

Seule F_{u_C} n'est pas définie. On considère donc la distribution des F_{u_C} induites par une fonction aléatoire C de distribution uniforme. On a bien une interprétation aléatoire F localement uniforme et homogène. On considère l'algorithme de résolution

$$R = ((\sigma_{u_x} \bowtie \sigma_{u_k}) \bowtie \sigma_{u_C}) \bowtie \sigma_{u_y}.$$

On calcule les $\mathrm{Sol}(V(S))$ pour les sous-termes S de R.

$$\begin{aligned}
\mathrm{Sol}_{u_x} &= 1 \\
\mathrm{Sol}_{u_K} &= \#K \\
\mathrm{Sol}(\{u_x, u_k\}) &= \#K \\
\mathrm{Sol}(\{u_x, u_k, u_C\}) &= \#K \\
\mathrm{Sol}_{u_y} &= 1 \\
\mathrm{Sol}(\{u_x, u_k, u_C, u_y\}) &= \#K/\#M.
\end{aligned}$$

On ne peut toutefois pas conclure $\mathrm{Comp}'(R) = \#K$. En effet, σ_{u_C} est un sous-terme de R tel que $\mathrm{Sol}_{u_C} = \#K.\#M$. On voit cependant que ce sous-terme n'est pas, dans la pratique, évalué. Avec une telle observation, on aurait tendance à modifier la structure du modèle de complexité. Ce n'est toutefois pas nécessaire, car si l'on considère des primitives de chiffrement usuelles

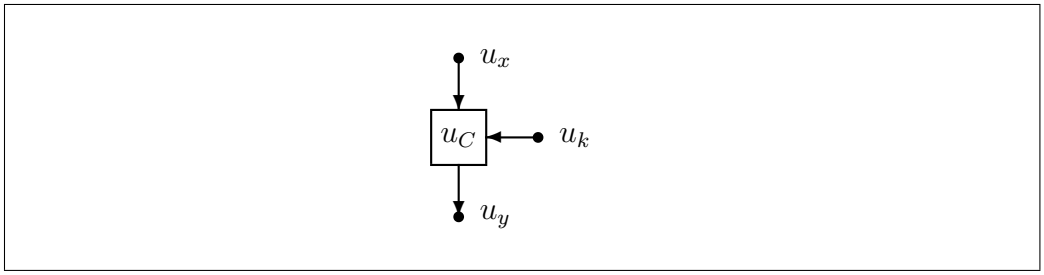


Figure 2.5 – Recherche Exhaustive.

définies par un réseau de calcul, on n'utilise jamais de boîtes de calcul aussi grandes que u_C. Avec le graphe d'équations de l'exemple 2.24, on a en fait

$$\mathrm{Comp}'(R) = \max\left(\#K, \max_u \mathrm{Sol}_u\right)$$

et Sol_u est souvent petit. Ce "défaut" du modèle a donc une portée limitée.

Exemple 2.33 (Attaque dans le milieu). Voici comme exemple d'application la notion bien connue d'"attaque dans le milieu". On suppose que l'on cherche à résoudre une équation du type

$$y = C'_{k_2}(C_{k_1}(x))$$

où k_1 et k_2 sont les inconnues d'un ensemble K_0. On pose $K = K_0^2$. On utilise le graphe d'équations suivant (Voir Fig. 2.6)

$$\begin{array}{rcl}
V & = & \{u_x, u_{k_1}, u_{k_2}, u_y, u_C, u_{C'}\} \\
E & = & \{u_x u_C, u_{k_1} u_C, u_C u_{C'}, u_{k_2} u_{C'}, u_{C'} u_y\} \\
\mathrm{Dom} & = & \begin{pmatrix} u_x u_C & u_{k_1} u_C & u_C u_{C'} & u_{k_2} u_{C'} & u_{C'} u_y \\ M & K_0 & M & K_0 & M \end{pmatrix} \\
\mathrm{Sol} & = & \begin{pmatrix} u_x & u_{k_1} & u_{k_2} & u_y & u_C & u_{C'} \\ 1 & \sqrt{\#K} & \sqrt{\#K} & 1 & \sqrt{\#K}.\#M & \sqrt{\#K}.\#M \end{pmatrix} \\
R_1 & = & (\sigma_{u_x} \bowtie \sigma_{k_1}) \bowtie \sigma_{u_{C_1}} \\
R_2 & = & (\sigma_{u_y} \bowtie \sigma_{k_2}) \bowtie \sigma_{u_{C_2}} \\
R & = & R_1 \bowtie R_2.
\end{array}$$

Hormis les sous-termes $\sigma_{u_{C_1}}$, $\sigma_{u_{C_2}}$ et R lui-même, on a $\mathrm{Sol}(V(S)) \leq \sqrt{\#K}$ pour les sous-termes S de R. La complexité de ce type d'attaque est donc de $\sqrt{\#K}$ pour trouver une clef dans K.

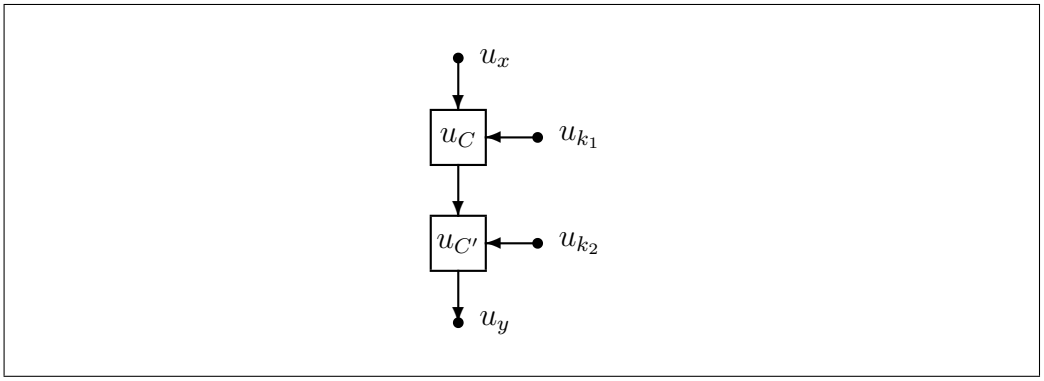


FIGURE 2.6 – Attaque dans le Milieu.

2.4.5 Application

Un exemple d'attaque effective qui utilise la notion de graphe d'équations est l'attaque de FFT Hash II.[20] Bien que cet algorithme ne soit pas une primitive de chiffrement, la notion de résolution de graphe d'équations est identique. FFT Hash II est une fonction de hachage proposée par Schnorr à la conférence Eurocrypt 92.[21]

2.4.6 Attaques par Projection

Lorsqu'un graphe d'équations ne peut pas être attaqué directement, on peut par des méthodes heuristiques le simplifier par la notion de "projection". Pour cela, on supprime tout simplement des arêtes en supposant que leur valeur a une distribution uniforme indépendante des autres, et pour toute interprétation f, on projette les ensembles f_u sur leur nouveau domaine. Un tel exemple d'expérience a été effectué sur l'algorithme DES, ce qui a permis d'obtenir la caractéristique utilisée par Matsui par un nouveau procédé. Cela a également permis d'améliorer sensiblement son attaque.

Concrètement, on définit le réseau de calcul de DES et l'on cherche, dans les deux registres du schéma de Feistel, à ne calculer qu'un faible nombre de bits. Pour tout choix possible de la position de ces bits, on peut supprimer tous les autres en leur supposant une distribution uniforme, ce qui procure aux boîtes de calcul un comportement aléatoire. On peut ensuite

[20] Cette attaque a été publiée dans les actes de Crypto 92 [161] (voir annexe A). En fait, cet article reporté en annexe A a historiquement été le point de départ pour fonder la notion de graphe d'équations.

[21] Cet article [143] fait suite à une proposition d'un précédent algorithme FFT Hashing non publié mais présenté à la conférence Crypto 91 [141] et qui a été cassé par Baritaud, Gilbert et Girault [20].

calculer la distribution des sorties du chiffrement en fonction de la distribution des entrées. On obtient en fait un opérateur linéaire représenté par une matrice stochastique. Si l'on cherche le choix des positions de bits qui maximise la distance entre cette matrice et celle dont tous les coefficients sont constants, on obtient précisément les bits utilisés dans la meilleure caractéristique linéaire de DES obtenus par Matsui. (Ce procédé retrouve donc la meilleure caractéristique linéaire par un moyen détourné.)

On peut ensuite exploiter ces positions par une attaque statistique qui utilise une fonction Σ linéaire, ou un test du χ^2.[22]

2.5 Conclusion

On a présenté les méthodes d'attaque générales usuelles :

- la cryptanalyse différentielle,
- la cryptanalyse linéaire,
- les attaques génériques sur les graphes d'équations (qui généralisent les recherches exhaustives et les attaques dans le milieu),
- les attaques statistiques obtenues par projection.

[22] Ces expériences publiées à la conférence ACM CCS 96 sont reportées en annexe C.

Chapitre 3

Sécurité du Chiffrement par Blocs

> "Perfect Secrecy *is defined by requiring of a system that after a cryptogram is intercepted by the enemy the* a posteriori *probabilities of this cryptogram representing various messages be identically the same as the* a priori *probabilities of the same message before the interception. It is shown that perfect secrecy is possible but requires, if the number of messages is finite, the same number of possible keys."*
> Claude E. Shannon
> In *Communication Theory of Secrecy Systems*, 1949.

La sécurité des algorithmes de chiffrement par blocs issus de la recherche publique était principalement empirique jusque dans les années 90. Tant qu'aucune attaque n'était connue, un algorithme était réputé sûr. L'apparition des méthodes générales d'analyse a cependant ouvert la voie à de nouveaux types de sécurité : on a dès lors cherché à démontrer la sécurité face à certains types d'attaques comme les attaques différentielles ou linéaires.

Dans ce chapitre, on présente les différents outils de preuve de sécurité. On définit la notion de "multipermutation" qui caractérise la diffusion parfaite. On montre comment déterminer des bornes inférieures de complexité pour des algorithmes génériques. On caractérise également la notion de non-linéarité optimale pour une boîte de calcul. Ceci permet de prouver une forme de sécurité heuristique. Enfin, on aborde les méthodes de preuve formelle, et notamment la notion de "décorrélation".

3.1 Sécurité Heuristique

Pour concevoir de nouveaux procédés de chiffrement symétriques, la notion de "sécurité heuristique" consiste en fait à ne pas faire les mêmes erreurs que les algorithmes déjà cassés. On cherche donc à rendre impossible des attaques déjà connues.

3.1.1 Emploi des Multipermutations

L'attaque contre l'algorithme Safer de la section 2.3.3 montre que la fonction

$$2\text{PHT}(x, y) = (2x + y \bmod 256, x + y \bmod 256)$$

laisse "fuir" un bit. On a en effet égalité entre le bit de poids faible de $2x+y \bmod 256$ et le bit de poids faible de y. De manière duale, il est possible de modifier l'entrée x (dans ce cas, sur son bit de poids fort) sans que la sortie $2x + y \bmod 256$ ne soit altérée : on a perdu un bit d'information.

Une propriété semblable permet d'attaquer une version simplifiée de la fonction MD4.[1]

De telles propriétés permettent de contrôler la diffusion d'informations au sein d'un réseau de calcul. Pour se prémunir contre ce type de faiblesse, on définit la notion de "multipermutation" qui caractérise la meilleure diffusion possible dans un tel réseau.[2]

Définition 3.1. *On considère une fonction f de D^p vers D^q. On dit que f est une multipermutation si pour deux $(p+q)$-uplets différents quelconques $(x, f(x))$ et $(y, f(y))$, il existe au moins $q+1$ entrées sur lesquelles ils prennent des valeurs différentes.*

En fait, si l'on considère une telle fonction comme définissant un code $C = \{(x, f(x)); x \in D^p\}$, la distance minimale est le nombre minimal non nul d'entrées pour lesquels deux $(p + q)$-uplets sont différents. Cette distance minimale est nécessairement inférieure à $q + 1$, car on peut construire x et y tels que seule une entrée est changée. Les multipermutations sont en fait les fonctions qui réalisent cette borne.

Voici quelques exemples élémentaires.

- $p = q = 1$: f est en fait une permutation de D.
- $p = 1$: f est en fait une famille $(f_1, \ldots, f_q)$ de q permutations de D.

[1] Voir à ce sujet [162] (annexe D).

[2] La notion de "multipermutation" est issue d'un travail sur les réseaux de calcul publié dans [145] et [146]. La définition s'est stabilisée dans [162] (voir annexe D).

- $p = 2$, $q = 1$: f définit en fait une structure de "pseudo-groupe" sur D, et sa table est un "carré latin".
- $p = 2$: f est une famille de q carrés latins deux-à-deux "orthogonaux".

Plus généralement, une (p, q)-multipermutation dur D est équivalente à un "tableau orthogonal" de type $((\#D)^p, p+q, \#D, p)$. De plus, on note qu'une multipermutation linéaire est équivalente à un code MDS.[3]

Avec de telles fonctions, des propriétés accidentelles qui mettent en œuvre peu d'entrées et de sorties d'une même fonction d'un réseau de calcul sont éliminées. Il est toutefois important de noter que ce critère de conception ne traite que la diffusion de calculs et ne constitue en rien un "critère de confusion" comme ceux qui sont présentés dans les sections suivantes. En effet, une multipermutation peut être une fonction linéaire (d'ailleurs, elles sont presque toujours construites à partir de telles fonctions) ce qui permet d'établir des caractéristiques différentielles ou linéaires dont le coefficient DP ou LP est 1. Ce que la multipermutation garantit, c'est que de telles caractéristiques rendront "actifs" un nombre maximal de sommets voisins.

Le problème principal réside dans la recherche de méthodes de construction de telles fonctions qui ne pénalisent pas les performances de l'algorithme. Pour cela, on emploie une multipermutation linéaire φ que l'on perturbe au niveau de toutes ses entrées et de toutes ses sorties par des permutations quelconques $\pi_1, \ldots, \pi_p$ et $\sigma_1, \ldots, \sigma_q$. On utilise ainsi

$$\begin{aligned} &f(x_1, \ldots, x_p) = \\ &\quad ((\sigma_1 \circ \varphi_1)(\pi_1(x_1), \ldots, \pi_p(x_p)), \ldots, (\sigma_q \circ \varphi_q)(\pi_1(x_1), \ldots, \pi_p(x_p))) \end{aligned}$$

où $\varphi = (\varphi_1, \ldots, \varphi_q)$. Il est évident qu'une telle fonction f est une multipermutation.

Pour construire une $(2, 1)$-multipermutation, on utilise une loi de groupe. Par exemple, sur $D = \mathbf{Z}_2^n$, on utilise

$$\varphi(y_1, y_2) = y_1 + y_2 \bmod 2^n$$

ou

$$\varphi(y_1, y_2) = y_1 \oplus y_2.$$

Pour construire une $(2, 2)$-multipermutation à partir d'un groupe $(D, +)$ (que l'on supposera abélien), on cherche des applications de la forme

$$f_i(x_1, x_2) = \sigma_i(\pi_1(x_1) + \pi_2(x_2)).$$

[3] On se référera à l'article [162] publié dans *Fast Software Encryption* 95 et reporté dans l'annexe D pour plus de détails.

On peut démontrer à partir du théorème de Hall-Paige[4] que de telles constructions sont possibles si et seulement si $\mathbf{Z}_2^2$ est un sous-groupe de D. En particulier, si D admet un sous-groupe isomorphe à $\mathbf{Z}_{2^k}$ et si $\#D/2^k$ est impair, une telle construction est impossible.

Dans le cas de la fonction Safer, le groupe $\mathbf{Z}_{256}$ rend donc la construction impossible. Par contre, on peut utiliser la construction suivante dans le groupe $\mathbf{Z}_2^n$ pour $n > 1$.

Théorème 3.2. *Soit σ une permutation de $\{1, \ldots, n\}$. Pour tout $x \in \mathbf{Z}_2^n$, on note x_σ l'élément de $\mathbf{Z}_2^n$ obtenu en permutant les entrées de x, soit $(x_\sigma)_i = x_{\sigma(i)}$. Soit $c \in \mathbf{Z}_2^n$. Pour tout $x \in \mathbf{Z}_2^n$, on note x AND c l'élément de $\mathbf{Z}_2^n$ obtenu par un "et" bit-à-bit, soit $(x \text{ AND } c)_i = x_i c_i$. On suppose que σ et c sont tels que*

$$c \text{ AND } c_\sigma \text{ AND } c_{\sigma\circ\sigma} \text{ AND } \ldots \text{ AND } c_{\sigma^n} = 0$$

et de même si l'on remplace c par son complémentaire bit-à-bit. (Autrement dit, l'itération de la permutation σ sur c fait passer un "0" et un "1" dans chaque entrée.) La fonction

$$\varphi(y_1, y_2) = ((c \text{ AND } (y_1)_\sigma) \oplus y_1 \oplus y_2, (y_1)_\sigma \oplus y_2)$$

est une multipermutation.

Une telle construction est évidemment très performante, car les instructions $\oplus$ et AND sont disponibles sur tous les microprocesseurs. La fonction de chiffrement CS-Cipher présentée en annexe G utilise une telle construction avec $n = 8$, $c = \texttt{55}_x$ et une rotation circulaire de un bit vers la gauche pour σ. L'intérêt d'utiliser $\texttt{55}_x$ est que l'on peut facilement inverser la fonction.

3.1.2 Critères Géométriques de Diffusion

Un réseau de calcul qui utilise des multipermutations permet de garantir une bonne diffusion de l'information. Sauf si les multipermutations sont particulières, on peut donc se ramener au problème de la sécurité du réseau de calcul muni d'une interprétation aléatoire, et donc aux propriétés géométriques du réseau.

Dans un exemple simple, on peut s'intéresser au problème de l'inversion de fonctions de hachage fondées sur le graphe FFT. On considère le réseau de calcul à 2^n entrées et r étages du graphe de la figure 1.6 (page 29). Avec les

[4] Ce théorème publié en 1955 dans [58] précise qu'il n'existe par de permutation π d'un groupe $(D, +)$ telle que $x \mapsto \pi(x) - x$ soit également une permutation lorsque le sous-groupe de Sylow d'ordre 2 de D est cyclique.

mêmes conventions que dans la section 1.3.1, si l'on note $u_{i_1,\ldots,i_n}$ les sommets d'entrée et $v_{i_1,\ldots,i_n}$ les sommets de sortie, et si l'on considère un ensemble D, on définit un réseau de calcul interprété $G = (V, E, \mathrm{Dom}, f)$ dans lequel $\mathrm{Dom}_a = D$ pour toute arête a. On définit ensuite une fonction de compression de D^{2^n} vers $D^{2^{n-1}}$ en mélangeant deux entrées x et y et en supprimant une sortie sur deux, soit

$$h(x, y) = \mathrm{Val}^G(t)|_{\{v_{i_1,\ldots,i_{n-1},0};i_1,\ldots,i_{n-1}\in\{0,1\}\}}$$

avec $t(u_{i_1,\ldots,i_{n-1},0}) = x_{i_1,\ldots,i_{n-1}}$ et $t(u_{i_1,\ldots,i_{n-1},1}) = y_{i_1,\ldots,i_{n-1}}$. Le problème de l'inversion étant donnée une valeur z et une valeur x, est de trouver une valeur y telle que $h(x, y) = z$. La recherche exhaustive porte ici sur $\#D^{2^{n-1}}$ possibilités.

Un tel problème définit un graphe d'équations $G_{n,m}$. On a le résultat suivant.

Théorème 3.3. *Pour des entiers n et m tels que $n < m \leq 2n - 1$ il existe un algorithme R de résolution de $G_{n,m}$ tel que*

$$\mathrm{Comp}'(R) = \#D^{2^{n-2}+2^{m-1-n}}.$$

Pour tout entier n, et tout algorithme de résolution R de $G_{n,2n-1}$, on a

$$\#D^{\frac{2}{3}2^{n-1}} \leq \min_R \mathrm{Comp}'(R) \leq \#D^{2^{n-1}}.$$

La borne $\#D^{\frac{2}{3}2^{n-1}}$ n'est sûrement pas la meilleure. On conjecture en fait que le meilleur algorithme de résolution est la recherche exhaustive, donc de complexité $\#D^{2^{n-1}}$. Cela signifie qu'il faut au moins $2n-1$ étages pour avoir une sécurité qui correspond à la recherche exhaustive.[5]

Ces résultats utilisent des notions de théorie des graphes. On démontre plus généralement que l'on peut obtenir une borne inférieure sur la complexité des algorithmes de résolution à partir des valeurs propres de la forme quadratique associée au graphe d'équations particuliers.[6]

Définition 3.4. *Un graphe d'équations $G = (V, E, \mathrm{Dom}, \mathrm{Sol})$ est dit "localement réversible" si pour tout sommet u on a*

$$\mathrm{Sol}_u = \prod_{uv\in E} \sqrt{\mathrm{Dom}_{uv}}.$$

[5] Ces résultats sont publiés dans [147] (voir annexe E).

[6] Le résultat suivant généralise en fait le théorème E.7 de l'annexe E p. 154.

En fait, lorsqu'un sommet u provient d'une boîte de calcul réversible, il existe une partition $\mathrm{In}_u \cup \mathrm{Out}_u$ qui définit les entrées et les sorties de u et telle que l'on ait

$$\mathrm{Sol}_u = \#\mathrm{Dom}(\mathrm{In}_u) = \#\mathrm{Dom}(\mathrm{Out}_u)$$

ce qui correspond à la notion de réversibilité locale.

Théorème 3.5. *Soit $G = (V, E, \mathrm{Dom}, \mathrm{Sol})$ un graphe d'équations localement réversible connexe. La forme quadratique Γ associée à G admet n valeurs propres*

$$0 = \lambda_1 < \lambda_2 \leq \ldots \leq \lambda_{\#V}.$$

Pour tout algorithme de résolution R, on a

$$\mathrm{Comp}'(R) \geq e^{\frac{2\lambda_2}{9}\#V}.$$

Preuve. Le fait que Γ admette des valeurs propres réelles est bien connu. Comme le graphe admet une seule solution, on a $\Gamma(V) = 0$. Comme la diagonale de Γ est dominante et positive, les valeurs propres sont positives, et le fait que λ_2 soit non nul provient du fait que G est connexe. Si $v_1, \ldots, v_{\#V}$ est une base orthonormale de vecteurs propres, on a, pour tout ensemble de sommets U,

$$\Gamma(U) = \sum_{i=1}^{\#V} \lambda_i (U \cdot v_i)^2 \geq \lambda_2((U \cdot U) - (U \cdot v_1)^2) \geq \lambda_2 \#U \left(1 - \frac{\#U}{\#V}\right)$$

car $U \cdot U = \#U$ et $U \cdot v_1 = \#U/\sqrt{\#V}$.

Pour tout entier c, il existe un sous-terme minimal S de R tel que

$$\#V(S) \geq c.$$

Au mieux, c'est la jointure de deux sous-termes d'au moins $c - 1$ sommets, on a donc $\#V(S) \leq 2c$. D'après l'inégalité suivante, on a

$$\Gamma(V(S)) \geq \min_{c \leq x \leq 2x} \lambda_2 x \left(1 - \frac{x}{\#V}\right)$$

donc

$$\log \mathrm{Comp}'(R) \geq \max_{0 \leq c \leq \#V} \min_{c \leq x \leq 2c} \lambda_2 x \left(1 - \frac{x}{\#V}\right) = \frac{2\lambda_2}{9}\#V$$

(voir le cas $c = x = \frac{\#V}{3}$). □

3.1.3 Critères Combinatoires de Confusion

Afin de garantir une sécurité heuristique face aux attaques différentielles et linéaires, une méthode consiste à rendre le coefficient DP ou LP de toute caractéristique petit. Comme ce coefficient est le produit de coefficients DP ou LP sur des sommets actifs, on construit l'algorithme de chiffrement suivant deux principes complémentaires :

- pour toute caractéristique telle que DP ou LP est non nul, le nombre de boîtes actives est grand ;
- pour tout sommet, la valeur maximale de $\mathrm{DP}(x, y)$ (pour x non nul) et $\mathrm{LP}(x, y)$ (pour y non nul) est petite.

Le premier critère est "naturellement" satisfait lorsque la diffusion de l'information au sein des boîtes de calcul est bonne. L'emploi de multipermutations l'assure. On peut également vérifier que ce critère est satisfait par des méthodes de comptage. Dans le cas de la fonction DES, les concepteurs se sont assurés que toute caractéristique différentielle aurait un nombre important de boîtes actives.[7] De même, Heys et Tavares[8] utilisent des propriétés géométriques d'un réseau particulier pour démontrer que le nombre de boîtes actives était nécessairement grand. C'est également l'approche adoptée dans la construction de la fonction CS-Cipher présentée en annexe G.

Le second critère s'obtient en étudiant les propriétés booléennes des boîtes de calcul. On définit

$$\mathrm{DP}^f_{\max} = \max_{a\neq 0, b} \mathrm{DP}^f(a, b) \tag{3.1}$$

$$\mathrm{LP}^f_{\max} = \max_{b\neq 0, a} \mathrm{LP}^f(a, b). \tag{3.2}$$

Les fonctions qui minimalisent les coefficients $\mathrm{DP}_{\max}$ et $\mathrm{LP}_{\max}$ sont rares et ont des propriétés mathématiques passionnantes. On a les résultats suivants.

Théorème 3.6. *Soit f une fonction de $\{0,1\}^p$ dans $\{0,1\}^q$. On a les bornes inférieures suivants, dont chacune définit une classe de fonctions qui réalisent les cas d'égalité.*[9]

- $\mathrm{DP}^f_{\max} \geq 2^{-q}$. *Cas d'égalité : fonctions "parfaitement non-linéaires" (PN).*

[7] Cela a été montré par Coppersmith dans un rapport publié en 1994, après la publication des méthodes d'attaques différentielles [38].

[8] Voir la thèse de Heys [63] et l'article [64] publié dans le *Journal of Cryptology*.

[9] Les résultats suivants ont été rassemblés dans l'article [35] publié au colloque Eurocrypt 94.

- $\mathrm{DP}^f_{\max} \geq 2^{1-p}$. *Cas d'égalité : fonctions "presque parfaitement non-linéaires" (APN).*[10]
- $\mathrm{LP}^f_{\max} \geq 2^{-p}$. *Cas d'égalité : fonctions "courbes" (B).*[11]
- $\mathrm{LP}^f_{\max} \geq 2^{1-p}\left(1 + \frac{(2^{q-p}-1)(2^{p-1}-1)}{2^q-1}\right)$. *Cas d'égalité : fonctions "presque courbes" (AB).*[12]

On a de plus les propriétés suivantes :

- *PN et B sont des propriétés équivalentes*[13] *et ne sont possibles que si et seulement si* $p \geq 2q$ *et* p *est pair*[14] *;*
- *AB n'est possible que si* $p = q$ *et* p *est impair, et entraîne APN sans que la réciproque soit vraie*[15] *;*
- *APN n'est possible que si* $(p, q) = (2, 1)$ *ou* $q \geq p$.

Un exemple de fonction AB est la fonction $f(x) = x^{1+2^k}$ dans $\mathrm{GF}(2^n)$ avec $1 < k < n$ et $\gcd(n, k) = 1$ (voir [111]).

Comme les fonctions APN ou AB sont rares et possèdent en général des propriétés mathématiques indésirables, on ne cherche pas à atteindre les bornes qu'elles réalisent, mais à les approcher. Par exemple, les auteurs de la fonction CAST Adams et Tavares[16] ont cherché à ce que des "parties" de la fonction réalisent des bornes intéressantes.

Dans la fonction CS-Cipher, on utilise des fonctions f et g telles que $p = q = 4$ et qui réalisent $\mathrm{DP}_{\max} \leq 2^{-2}$ et $\mathrm{LP}_{\max} \leq 2^{-2}$. Ces fonctions servent à construire une permutation P telle que $p = q = 8$, $\mathrm{DP}_{\max} \leq 2^{-4}$ et $\mathrm{LP}_{\max} \leq 2^{-4}$. CS-Cipher utilise des $(2, 2)$-multipermutations, ce qui impose qu'une boîte active est nécessairement adjacente à au moins trois autres boîtes actives. Ce critère, et la structure géométrique du réseau de calcul de CS-Cipher permettent ainsi de démontrer que toute caractéristique possède au moins 72 boîtes actives. Le coefficient DP ou LP théorique de toute caractéristique est donc nécessairement inférieur à 2^{-288}, ce qui garantit la sécurité heuristique face aux attaques différentielles et linéaires.

[10] Voir Nyberg-Knudsen [115].
[11] Voir Rothaus [133].
[12] Voir Chabaud-Vaudenay [35].
[13] Voir Meier-Staffelbach[100].
[14] Voir Nyberg [110].
[15] Voir Chabaud-Vaudenay [35].
[16] Voir la thèse d'Adams [11] et l'article [12] publié dans le *Journal of Cryptology*. On pourra également se référer à l'analyse de sécurité face aux attaques différentielles par Lee, Heys et Tavares [83].

3.2 Sécurité Prouvée

Dans cette section, on présente les différentes méthodes qui conduisent à une sécurité prouvée dans le domaine du chiffrement symétrique. Ces approches n'ont aucun lien logique apparent, mais seront rassemblées dans la théorie de la décorrélation de la section suivante.

3.2.1 Approche de Shannon

On ne peut parler de preuve de sécurité du chiffrement symétrique sans mentionner le théorème de Shannon qui caractérise la notion de "sécurité parfaite".[17] Celui-ci utilise la notion d'"entropie". Pour une variable aléatoire X, on définit

$$H(X) = -\sum_x \Pr[X = x] \log_2 \Pr[X = x]$$

avec la convention $0 \log_2 0 = 0$. On définit également l'entropie conditionnelle $H(X/Y) = H(X,Y) - H(Y)$.

Suivant le formalisme de Shannon, on considère qu'une fonction de chiffrement C est destinée à ne chiffrer qu'un seul message X (que l'on peut considérer comme étant la concaténation de tous les messages à transmettre. On note cependant que l'on n'a pas de fonction véritablement définie sur chacun de ces "morceaux de message").

On considère donc une fonction de chiffrement aléatoire de distribution donnée C (autrement dit, une fonction définie par une clef aléatoire suivant un procédé donné), un message clair X et un message chiffré $Y = C(X)$. On formalise ainsi la notion de sécurité parfaite.

Définition 3.7. *Une fonction de chiffrement aléatoire C assure une "confidentialité parfaite" sur un message aléatoire X si l'on a*

$$H(X/C(X)) = H(X).$$

Intuitivement, cela signifie que la donnée de l'information $C(X)$ n'apporte aucune aide pour déterminer une quelconque information nouvelle sur X. Le résultat de Shannon est le suivant.

Théorème 3.8 (Shannon 1949). *Si une fonction de chiffrement aléatoire C assure une confidentialité parfaite sur un message aléatoire X, on a*

$$H(C) \geq H(X).$$

[17] Les résultats de cette section ont été publiés en 1949 [150].

Cela signifie notamment que la clef qui définit C doit être au moins aussi longue que X.

Par exemple, si l'on considère une structure de groupe sur l'ensemble de messages $\mathcal{M}$, pour une clef aléatoire K de distribution uniforme sur $\mathcal{M}$, la fonction

$$C(X) = X + K$$

assure une confidentialité parfaite de X. C'est l'idée du chiffrement de Vernam.[18]

3.2.2 Approche de Carter-Wegman

La notion de "fonction universelle" d'ordre 2 a été définie par Carter et Wegman et appliquée à la notion de code d'authentification de message (MAC).[19] Suivant cette notion, une fonction aléatoire C de $\mathcal{M}_1$ vers $\mathcal{M}_2$ est "ϵ-presque fortement universelle d'ordre 2" si pour deux éléments distincts quelconques x_1 et x_2 de $\mathcal{M}_1$ et pour deux éléments quelconques y_1 et y_2 de $\mathcal{M}_2$, on a

$$\Pr[C(x_1) = y_1, C(x_2) = y_2] \leq \frac{1}{\#\mathcal{M}_2^2} + \epsilon.$$

On suppose que l'on dispose qu'une fonction de hachage h_{K_0} d'un ensemble $\mathcal{M}$ vers un groupe G définie par une clef secrète aléatoire K_0, et d'une suite de clefs $K_1, \ldots, K_d$ aléatoires de distribution uniforme dans le groupe G. On suppose en outre ces clefs indépendantes. Pour d messages $m_1, \ldots, m_d$ dans $\mathcal{M}$, on définit leurs codes d'authentification

$$c_i = K_i + h_{K_0}(m_i).$$

Dans un modèle d'attaque, la suite des couples (m_i, c_i) est transmise au travers d'un canal de communication peu sûr, et l'attaquant peut à loisir modifier, supprimer, échanger des messages. Son objectif est que la suite (m'_i, c'_i) reçue par le destinataire soit différente de la suite interceptée (et de même longueur d), et que la relation

$$c'_i = K_i + h_{K_0}(m'_i)$$

[18] Ce système publié en 1926 [181] était considéré comme sûr. C'est en fait Shannon qui a su introduire le formalisme nécessaire pour le démontrer.

[19] Cette notion a été définie dans le *Journal of Computer and System Sciences* en 1979 [34] et étendue à la notion de MAC en 1981 [183].

soit vérifiée. On a le résultat suivant.[20]

Théorème 3.9 (Wegman-Carter 1981). *Avec l'algorithme de MAC précédent, si un attaquant ne dispose d'aucune information sur les clefs et si h_K est ϵ-presque fortement universelle d'ordre 2, sa probabilité de succès sera inférieure à ϵ quelle que soit sa stratégie.*

La notion de fonction universelle se généralise naturellement. On adoptera par la suite la notion de "décorrélation".

3.2.3 Approche de Luby-Rackoff

Un autre résultat important, dû à Luby et Rackoff, démontre que la construction de Feistel n'introduit pas de faiblesse générique. Ils démontrent que si les fonctions d'étage sont aléatoires, on obtient alors (avec un minimum de trois étages) une fonction de chiffrement aléatoire.

Il faut bien comprendre que l'objectif d'un procédé de chiffrement est de ressembler à une transformation aléatoire sur la distribution de la clef. Comme les clefs ont en général une longueur faible, le nombre de permutations engendrées a en général une densité trop faible pour être aléatoire. D'un point de vue formel, on a ici un problème de "dérandomisation".

Pour formaliser la notion de "ressemblance à une fonction aléatoire", on s'intéresse au modèle d'attaque par distingueur. On compare en fait la fonction de chiffrement réelle (pseudo-aléatoire) à une fonction de chiffrement idéale (aléatoire), et l'on cherche, par un jeu de questions et de réponses à les distinguer. Pour montrer qu'elles se ressemblent, on majore en fait l'avantage obtenu par tout distingueur.

On précise quelques définitions qui ont déjà été motivées dans la section 2.1.3.

Définition 3.10. *Etant donnés deux ensembles $\mathcal{M}_1$ et $\mathcal{M}_2$, un "distingueur" pour une fonction de $\mathcal{M}_1$ vers $\mathcal{M}_2$ est une machine de Turing probabiliste $\mathcal{A}$ qui dispose d'un oracle qui pour toute requête de $\mathcal{M}_1$ envoie une réponse dans $\mathcal{M}_2$ et qui, après un certain nombre de calculs, fournit une réponse binaire : "1" ou "0". On caractérise le distingueur par*

[20] Ce résultat a été amélioré par Krawczyk qui a démontré qu'il suffisait que h_K soit "ϵ-presque Δ-uniforme", soit que pour tout $x \neq y$ et tout a, on a

$$\Pr[h_K(x) - h_K(y) = a] \leq \frac{1}{\#G} + \epsilon$$

et non ϵ-presque fortement universelle d'ordre 2. Ce résultat a été présenté au colloque Crypto 94 [77].

- *le nombre de requêtes d à l'oracle,*
- *le temps de calcul t,*
- *le fait qu'il prépare toutes ses requêtes à l'avance (le distingueur est "non-adaptatif") ou qu'il effectue des requêtes en fonction des réponses aux précédentes (le distingueur est "adaptatif").*

Etant donnée une fonction aléatoire F de $\mathcal{M}_1$ vers $\mathcal{M}_2$, on définit $E(\mathcal{A}^F)$ comme étant la probabilité que $\mathcal{A}$ fournisse la réponse "1" lorsque l'oracle implémente la fonction F. La probabilité porte sur la distribution de F et du comportement aléatoire de $\mathcal{A}$. Etant données deux fonctions aléatoires F et G, l'avantage de $\mathcal{A}$ pour distinguer F et G est

$$\mathrm{Adv}_{\mathcal{A}}(F,G) = \left| E(\mathcal{A}^F) - E(\mathcal{A}^G) \right|.$$

Le théorème de Luby-Rackoff s'énonce ainsi.[21]

Théorème 3.11 (Luby-Rackoff 1988). *Pour un ensemble $\mathcal{M} = \{0,1\}^m$, on considère trois fonctions aléatoires F_1, F_2 et F_3 indépendantes et de distribution uniforme de $\{0,1\}^{\frac{m}{2}}$ vers $\{0,1\}^{\frac{m}{2}}$. On considère également une permutation aléatoire C^* sur $\mathcal{M}$ de distribution uniforme. Pour tout distingueur $\mathcal{A}$ entre C^* et $\Psi(F_1,F_2,F_3)$ limité à d requêtes, on a*

$$\mathrm{Adv}_{\mathcal{A}}(\Psi(F_1,F_2,F_3),C^*) \leq \frac{d^2}{2^{\frac{m}{2}}}.$$

En supposant que la clef procure aux fonctions d'étage des distributions uniformes et indépendantes, on obtient donc une sécurité relative garantie tant que la fonction de chiffrement est utilisée un nombre de fois négligeable devant $2^{\frac{m}{4}}$.

3.2.4 Approche de Nyberg-Knudsen

Dans cette section, on s'intéresse à démontrer la sécurité face aux attaques différentielles et linéaires. On définit pour cela une nouvelle notation. Etant donnée une fonction de chiffrement C_K définie par une clef aléatoire K de distribution donnée, on note

$$\begin{aligned} \mathrm{EDP}^C(a,b) &= \mathop{E}_K\left(\mathrm{DP}^{C_K}(a,b)\right) \\ \mathrm{ELP}^C(a,b) &= \mathop{E}_K\left(\mathrm{LP}^{C_K}(a,b)\right). \end{aligned}$$

[21] Il fut publié dans le *Journal on Computing* [86]. De nombreuses généralisations ont été effectuées par la suite, notamment par Patarin [118, 119, 121] et Pieprzyk [122].

On considère ainsi la valeur moyenne des coefficients DP et LP sur la distribution de la clef. On définit de même

$$\mathrm{EDP}^C_{\max} = \max_{a\neq 0,b} \mathrm{EDP}^C(a,b) \tag{3.3}$$

$$\mathrm{ELP}^C_{\max} = \max_{b\neq 0,a} \mathrm{ELP}^C(a,b). \tag{3.4}$$

Pour cela, on cherche à rendre $\mathrm{DP}^G_{\max}$ et $\mathrm{LP}^G_{\max}$ minimal pour l'ensemble de la fonction de chiffrement G. La méthode de Nyberg et Knudsen consiste à majorer les coefficients $\mathrm{EDP}^C_{\max}$ et $\mathrm{ELP}^C_{\max}$ par une construction dédiée. Le théorème suivant a été démontré (dans une forme moins forte) par Nyberg et Knudsen, puis par Aoki et Ohta.[22]

Théorème 3.12. *On considère des permutations f_i, $i = 1, 2, 3$, de $\{0,1\}^{\frac{m}{2}}$ telles que $\mathrm{DP}^{f_i}_{\max} \leq p$ pour tout i. On pose*

$$F_i(x) = f_i(x \oplus K_i)$$

où $K = (K_1, K_2, K_3)$ est une clef aléatoire de distribution uniforme sur $\{0,1\}^{\frac{3m}{2}}$. On a $\mathrm{EDP}_{\max}(\Psi(F_1,F_2,F_3)) \leq p^2$. De même, si l'on a $\mathrm{LP}^{f_i}_{\max} \leq p$ pour tout i, on obtient $\mathrm{ELP}_{\max}(\Psi(F_1,F_2,F_3)) \leq p^2$.

Un tel résultat de borne supérieure sur $\mathrm{EDP}_{\max}$ et $\mathrm{ELP}_{\max}$ permet de prouver la sécurité face aux attaques différentielles et linéaires. Contrairement aux approches heuristiques qui se contentent de majorer les coefficients DP et LP des caractéristiques, on s'intéresse ici à l'effet cumulé de toutes les caractéristiques (obtenu par exemple dans l'équation (2.6)) et sans approximations.

A l'origine, Nyberg et Knudsen ont utilisé ce résultat pour construire des fonctions de chiffrement dans lesquelles les f_i sont des fonctions presque courbes (AB). Les exemples utilisés introduisent cependant d'autres faiblesses. En fait, les fonctions AB utilisées ont pour degré algébrique 2 ce qui permet d'envisager d'autres attaques (de type algébrique).[23]

[22] Il a été pour la première fois présenté à Crypto 92, puis dans le *Journal of Cryptology* [114, 115]. Dans ces articles, Nyberg et Knudsen n'envisagent que les coefficients DP (la cryptanalyse linéaire n'était pas encore publiée) et obtiennent une borne de $2p^2$. Nyberg [112] a ensuite considéré les coefficients LP, et la borne a ensuite été améliorée par Aoki et Ohta [18]. Dans l'article original de Nyberg et Knudsen, on démontre également que la borne $2p^2$ reste valable avec quatre étages lorsque les f_i ne sont pas des permutations. On mentionne également les résultats de Matsui [92, 93] qui appliquent ce type de résultat à des constructions voisines de celle du schéma de Feistel. Un récent article de Nyberg [113] tente également de relancer la généralisation systématique de ces résultats à d'autres types de construction.

[23] Une attaque due à Jakobsen et Knudsen contre l'un de ces exemples a été exhibée dans [67].

Le théorème 3.12 peut également être utilisé de manière récursive pour construire une suite C^n de schémas de Feistel sur des blocs de longueur $2^n.m_0$, tels que $\mathrm{EDP}_{\max}(C^n) \leq p^{2^n}$ et $\mathrm{ELP}_{\max}(C^n) \leq p^{2^n}$. Cette idée est à l'origine de la fonction de chiffrement MISTY.[24] Cette construction n'élimine cependant pas le problème du degré algébrique faible.

3.3 Théorie de la Décorrélation

Dans cette section, on présente quelques résultats issus de la théorie de la décorrélation. On se contente ici de présenter des résultats sans preuve.[25]

3.3.1 Définitions

La notion de décorrélation est analogue aux généralisations classiques de la notion de fonction universelle. On utilise les définitions et notations suivantes.

Définition 3.13. *Etant donnés une fonction aléatoire F d'un ensemble $\mathcal{M}_1$ vers un ensemble $\mathcal{M}_2$ et un entier naturel d, on définit la "matrice de décorrélation d'ordre d" que l'on note $[F]^d$ la matrice dont les lignes sont indexées par une famille $x = (x_1, \ldots, x_d)$ de d éléments de $\mathcal{M}_1$ et les colonnes sont indexées par une famille $y = (y_1, \ldots, y_d)$ de d éléments de $\mathcal{M}_2$, et telle que*

$$[F]^d_{x,y} = \Pr[F(x_i) = y_i; i = 1, \ldots, d].$$

Intuitivement, la décorrélation d'une fonction aléatoire F est la distance de $[F]^d$ à $[F^*]^d$ pour une fonction F^* de référence (dans ce cas, une fonction de distribution uniforme). De même, la décorrélation d'une permutation aléatoire C est la distance de $[C]^d$ à $[C^*]^d$ pour une permutation C^* de distribution uniforme. La notion de décorrélation ne peut cependant pas être définie formellement autrement que par un procédé de comparaison qui dépend du choix de la notion de distance.

Définition 3.14. *Etant donnés deux fonctions aléatoires F et G d'un ensemble $\mathcal{M}_1$ vers un ensemble $\mathcal{M}_2$, un entier naturel d et une distance D sur l'ensemble de matrices $\mathbf{R}^{\mathcal{M}_1^d \times \mathcal{M}_2^d}$, on appelle $D([F]^d, [G]^d)$ la "distance de décorrélation d'ordre d entre F et G". Lorsqu'elle est nulle, on dit que F et G ont la même décorrélation.*

[24] Cette fonction a été développée par Mistubishi Electronic Corporation par une équipe coordonnée par Matsui. Leurs auteurs sont à l'origine de cette idée d'utiliser le théorème de Nyberg-Knudsen de manière récursive. Leurs résultats ont été présentés au colloque *Fast Software Encryption* 97 [93].

[25] On pourra se référer à l'annexe F pour plus de détails.

Si G admet une distribution uniforme, on appelle $D([F]^d, [G]^d)$ le "biais de décorrélation d'ordre d de la fonction F" que l'on note $\mathrm{DecF}^d(F)$. *Si F et G sont bijectives et si la distribution de G est uniforme parmi l'ensemble des permutations, on appelle $D([F]^d, [G]^d)$ le "biais de décorrélation d'ordre d de la permutation F" que l'on note* $\mathrm{DecP}^d(F)$. *Lorsqu'un biais de décorrélation est nul, on dit que la décorrélation est "parfaite".*

Dans toute la suite, on utilisera la distance induite par la norme $|||.|||_\infty$ définie par

$$|||A|||_\infty = \max_x \sum_y |A_{x,y}| \tag{3.5}$$

et que l'on notera $||.||$ par commodité.[26]

Exemple 3.15. La fonction de chiffrement de Vernam définie sur un groupe G par $C(x) = x + K$ où K admet une distribution uniforme admet une décorrélation d'ordre 1 parfaite. En revanche, on a

$$\mathrm{DecP}^2(C) = 2 + \frac{1}{\#G - 1} \approx 2.$$

Exemple 3.16. La fonction de chiffrement $C(x) = A.x + B$ définie par une clef $K = (A, B)$ aléatoire telle que $A \neq 0$ et de distribution uniforme sur un corps admet une décorrélation (de permutation) d'ordre 2 parfaite.

Voici un autre exemple fondamental.

Exemple 3.17. Pour la fonction $F(x) = ((A.x + B) \bmod p) \bmod 2^m$ définie sur $\{0,1\}^m$ par une clef $K = (A, B) \in \{0,1\}^{2m}$ de distribution uniforme et un nombre premier $p \geq 2^m$, on a

$$\mathrm{DecF}^2(F) = 4\epsilon + 2\epsilon^2$$

où $p = 2^m(1+\epsilon)$. Par exemple, pour $m = 64$ et $p = 2^{64}+13$, on a $\mathrm{DecF}^2(F) \approx 2^{-58.3}$ qui est très petit.

[26] Cette norme a permis d'énoncer des résultats de sécurité intéressants. D'autres normes ont été considérées, notamment la norme euclidienne

$$||A||_2 = \sqrt{\sum_{x,y} A_{x,y}^2}$$

mais celle-ci a donné des résultats moins bons. On pourra se référer à l'article [176] présenté au colloque SAC 98.

3.3.2 Résultats de Sécurité Revisités

On effectue ici le lien avec les résultats de la section 3.2 pour montrer que la notion de décorrélation les rassemble.

Le théorème 3.9 de Wegman-Carter s'énonce naturellement en termes de décorrélation d'ordre 2, mais au moyen d'une autre norme que $|||.|||_\infty$.

Le théorème 3.8 de Shannon ne se transpose pas immédiatement, car l'entropie s'exprime difficilement en terme de décorrélation. Cependant, on démontre que la propriété d'assurer une confidentialité parfaite pour une variable aléatoire de distribution uniforme X est équivalente à la décorrélation parfaite à l'ordre 1. La notion d'ordre de décorrélation permet également d'énoncer des résultats de sécurité parfaite pour des fonctions C utilisées plusieurs fois (et non, dans le cas du théorème de Shannon, pour une fonction utilisée une seule fois pour l'ensemble des messages à chiffrer).

Théorème 3.18. *Soit une fonction de chiffrement C admettant une décorrélation parfaite à l'ordre d. Soient $x_1, \ldots, x_{d-1}$ des messages quelconques, et X un message aléatoire différent des x_i. On a*

$$H(X/C(x_1), \ldots, C(x_{d-1}), C(X)) = H(X).$$

Cela signifie que si l'on utilise d fois la fonction C et que les $d-1$ premiers messages sont compromis, le dernier message est toujours parfaitement confidentiel, pourvu qu'il soit différent des autres.

Le théorème 3.11 de Luby-Rackoff peut s'énoncer ainsi en termes de décorrélation.

Théorème 3.19. *Pour un ensemble $\mathcal{M} = \{0,1\}^m$ (pour un entier m pair), on considère trois fonctions aléatoires F_1, F_2 et F_3 indépendantes de $\{0,1\}^{\frac{m}{2}}$ vers $\{0,1\}^{\frac{m}{2}}$ et de décorrélation parfaite à l'ordre d. On a*

$$\mathrm{DecP}^d(\Psi(F_1, F_2, F_3), C^*) \leq \frac{2d^2}{2^{\frac{m}{2}}}.$$

Si ce résultat n'est pas très parlant (puisque le biais de décorrélation n'est pas une borne inférieure de complexité d'une attaque en lui-même), il peut s'éclairer à la lumière du théorème suivant qui montre l'équivalence entre l'avantage du meilleur distingueur non adaptatif et la décorrélation au sens de la norme que l'on a choisie.

Théorème 3.20. *Si l'on considère deux fonctions aléatoires F et G de distributions données, le meilleur distingueur $\mathcal{A}$ non-adaptatif limité à d requêtes entre F et G admet pour avantage*

$$\mathrm{Adv}_{\mathcal{A}}(F, G) = \frac{1}{2}||[F]^d - [G]^d||.$$

Le théorème 3.19 signifie donc que le meilleur distingueur non-adaptatif entre $\Psi(F_1, F_2, F_3)$ et C^* a un avantage inférieur à $d^2/2^{\frac{m}{2}}$. Ce résultat est moins fort que le théorème 3.11 d'origine, car on se restreint aux attaques non-adaptatives, mais on peut énoncer un résultat semblable en employant une autre norme.

Enfin, il convient de chercher l'analogue du théorème 3.12 de Nyberg-Knudsen. En fait, on montre dans la section suivante que la décorrélation à l'ordre 2 garantit la sécurité face aux attaques différentielles et linéaires, et l'on montre comment la fonction de chiffrement hérite de la décorrélation de chaque fonction d'étage d'un schéma de Feistel.

Théorème 3.21. *On considère l'ensemble $\mathcal{M} = \{0,1\}^m$ (pour m pair), et deux entiers naturels $r \geq 3$ et d. Pour r fonctions aléatoires indépendantes $F_1, \ldots, F_r$ sur $\{0,1\}^{\frac{m}{2}}$, si l'on a $\mathrm{DecF}^d(F_i) \leq \epsilon$ pour tout i, alors on a*

$$\mathrm{DecP}^d(\Psi(F_1, \ldots, F_r)) \leq \left(3\epsilon + 3\epsilon^2 + \epsilon^3 + \frac{2d^2}{2^{\frac{m}{2}}}\right)^{\left\lfloor \frac{r}{3} \right\rfloor}$$

On peut donc utiliser les fonctions de l'exemple 3.17 pour construire des fonctions de chiffrement suffisamment décorrélées à l'ordre 2. Par exemple, la fonction DFC présentée en annexe H utilise $m = 128$, $r = 8$, $d = 2$ et $\epsilon \approx 2^{-58.3}$, donc on a $\mathrm{DecP}^2(\mathrm{DFC}) \leq 2^{-113.3}$.

3.3.3 Résistance contre des Classes d'Attaques

Les coefficients EDP et ELP d'une fonction de chiffrement aléatoire C s'expriment naturellement en fonction des coefficients de sa matrice de décorrélation d'ordre 2. Cela conduit aux majorations suivantes.

Théorème 3.22. *Pour une fonction de chiffrement aléatoire C sur $\{0,1\}^m$, on a*

$$\begin{aligned} \mathrm{EDP}^C_{\max} &\leq \frac{1}{2}\mathrm{DecP}^2(C) + \frac{1}{2^m - 1} \\ \mathrm{ELP}^C_{\max} &\leq 2\mathrm{DecP}^2(C) + \frac{1}{2^m - 1}. \end{aligned}$$

La décorrélation à l'ordre 2 est donc adaptée au même titre que les coefficients $\mathrm{EDP}_{\max}$ et $\mathrm{ELP}_{\max}$ pour garantir une sécurité face aux attaques différentielles et linéaires.

Les techniques utilisées dans cette théorie permettent d'ailleurs de préciser le rôle exact de ces coefficients une fois les attaques différentielles et linéaires correctement modélisées. On définit un "distingueur différentiel" par une différence d'entrée a, une différence de sortie b, une complexité n et le programme suivant.

1. pour $i = 1$ jusqu'à n, faire
   (a) tirer un X aléatoire
   (b) effectuer les requêtes X et $X \oplus a$ pour obtenir de l'oracle les réponses Y et Y'
   (c) si $Y \oplus Y' = b$, s'arrêter avec le résultat "1"
2. fournir le résultat "0"

On a le résultat suivant.

Théorème 3.23. *Etant donnés le distingueur précédent de complexité n, et deux fonctions de chiffrement aléatoires C et C^* sur $\{0,1\}^m$, C^* étant de distribution uniforme, on a*

$$\mathrm{Adv}(C, C^*) \leq n. \max\left(\frac{1}{2^m - 1}, \mathrm{EDP}^C(a, b)\right).$$

De même, on définit un distingueur linéaire par a, b, n, un intervalle I et le programme suivant.

1. initialiser un compteur c
2. pour $i = 1$ jusqu'à n, faire
   (a) tirer un X aléatoire
   (b) effectuer la requête X pour obtenir de l'oracle la réponses Y
   (c) faire $c \leftarrow c + ((a \cdot X) \oplus (b \cdot Y))$
3. si $c \in I$, fournir le résultat "1", sinon, fournir le résultat "0"

Et l'on a le résultat suivant.

Théorème 3.24. *Etant donnés le distingueur précédent de complexité n, et deux fonctions de chiffrement aléatoires C et C^* sur $\{0,1\}^m$, C^* étant de distribution uniforme, on a*

$$\lim_{n \to +\infty} \frac{\mathrm{Adv}(C, C^*)}{n^{\frac{1}{3}}} \leq 9.3 \left(\max\left(\frac{1}{2^m - 1}, \mathrm{ELP}^C(a, b)\right)\right)^{\frac{1}{3}}.$$

Dans la pratique, cette limite est très rapide si bien que les valeurs utilisées de n (par exemple, $n > 2^{32}$) peuvent être considérées comme infinies.

On montre également la résistance face à un type d'attaques plus général : les attaques itératives. Formellement, un distingueur itératif d'ordre d se définit par un programme qui respecte la structure suivante.

1. pour $i = 1$ jusqu'à n, faire

   (a) effectuer d requêtes $X^i = (X_1^i, \ldots, X_d^i)$ pour obtenir de l'oracle O les réponses $Y^i = (O(X_1^i), \ldots, O(X_d^i))$
   (b) définir $T_i = 0$ ou 1
2. si $(T_1, \ldots, T_n) \in A$ fournir le résultat "1", sinon, fournir le résultat "0"

Ce programme est défini par une complexité n, une procédure de décision pour T_i, et un test final A. L'idée importante est que le distingueur n'a le droit de retenir qu'une information binaire entre deux jeux de d requêtes. Pour que ce principe soit respecté, il faut supposer que les itérations sont indépendantes, donc que la sélection des d requêtes d'un jeu ne dépend pas du résultat des autres. On supposera de manière plus restrictive que le distingueur est non-adaptatif (donc que le choix des d requêtes est également non-adaptatif) et que toutes les itérations sont identiques : les requêtes sont choisies avec la même distribution. Enfin, on supposera que la procédure de décision pour T_i ne dépend que de X^i et Y^i. En fait, on pourra définir la distribution de T_i en fonction de X^i et Y^i par son espérance $\mathcal{T}(X^i, Y^i)$. On a

$$\forall i \quad \mathcal{T}(x, y) = \Pr[T_i = 1 / X^i = x, Y^i = y].$$

On adopte donc la définition suivante.

Définition 3.25. *Pour deux entiers naturels d et n et un ensemble $\mathcal{M}$, un "distingueur itératif" d'ordre d et de complexité n est défini par la donnée*

- *d'une distribution $\mathcal{D}_X$ sur $\mathcal{M}^d$,*
- *d'une fonction $\mathcal{T}$ de $\mathcal{M}^{2d}$ vers $[0, 1]$,*
- *d'un sous-ensemble A de $\{0, 1\}^n$.*

Le programme suivant définit le distingueur.

1. *pour $i = 1$ jusqu'à n, faire*
   (a) sélectionner d requêtes $X \leftarrow (X_1, \ldots, X_d)$ aléatoires suivant la distribution $\mathcal{D}_X$
   (b) obtenir les réponses $Y \leftarrow (O(X_1), \ldots, O(X_d))$ de l'oracle O
   (c) avec probabilité $\mathcal{T}(X, Y)$, faire $T_i \leftarrow 1$ (sinon, $T_i = 0$)
2. *si $(T_1, \ldots, T_n) \in A$ fournir le résultat "1", sinon, fournir le résultat "0"*

On peut exprimer les attaques différentielles comme un cas particulier d'attaque itérative d'ordre 2. Pour cela, $\mathcal{D}_X$ est la distribution $(X, X \oplus a)$ pour un X de distribution uniforme. La fonction $\mathcal{T}((x_1, x_2), (y_1, y_2))$ vaut 1

lorsque l'on a $x_1 \oplus x_2 = a$ et $y_1 \oplus y_2 = b$. Enfin, l'ensemble A est l'ensemble de tous les n-uplets $(t_1, \ldots, t_n)$ sauf $(0, \ldots, 0)$.

Les attaques linéaires sont un cas particulier d'attaque itérative d'ordre 1. La distribution $\mathcal{D}_X$ est ici uniforme. On a $\mathcal{T}(x, y) = (a \cdot x) \oplus (b \cdot y)$, et l'ensemble A est un ensemble du type

$$A = \left\{(t_1, \ldots, t_n); \left| t_1 + \ldots + t_n - \frac{n}{2} \right| \geq \lambda n \right\}$$

pour un certain coefficient λ. On a le résultat général suivant.

Théorème 3.26. *Soit une fonction de chiffrement aléatoire C sur un ensemble $\mathcal{M}$. Pour deux entiers naturels d et n, on considère un distingueur itératif $\mathcal{A}$ d'ordre d et de complexité n qui utilise la distribution $\mathcal{D}_X$. On note $\epsilon = \mathrm{DecP}^{2d}(C)$ et*

$$\delta = \Pr_{\substack{X \in \mathcal{D}_X \\ X' \in \mathcal{D}_X}} [\exists i, j \ X_i = X'_j]$$

où X et X' sont indépendantes. Si l'on cherche à distinguer C d'une permutation aléatoire C^* de $\mathcal{M}$ de distribution uniforme, l'avantage est inférieur à

$$3 \left(\left(2\delta + \frac{5d^2}{2\#\mathcal{M}} + \frac{3\epsilon}{2} \right) n^2 \right)^{\frac{1}{3}} + \frac{n\epsilon}{2}$$

si $2d \leq \#\mathcal{M}$.

Par exemple, pour une attaque itérative d'ordre 1 qui utilise une distribution $\mathcal{D}_X$ uniforme, et pour $\mathrm{DecP}^2(C) \leq 1/\#\mathcal{M}$, l'avantage est négligeable lorsque n est petit devant $\sqrt{\#\mathcal{M}}$.

La théorie est encore jeune, et ce résultat demande à être encore amélioré. Ce type de résultat illustre en fait les possibilités offertes par la notion de décorrélation pour démontrer la sécurité face à une classe d'attaques bien définie.

3.4 Conclusion

L'expérience des attaques connues pour le chiffrement symétrique suggère en fait un retour aux sources : il faut assurer une bonne diffusion et une bonne confusion dans les calculs. On peut cependant modéliser ces notions intuitives par des critères quantitatifs.

La qualité de la diffusion au niveau local des boîtes dépend en fait de la distance minimale (au sens de la théorie des codes) qui correspond à ces boîtes de calcul. La diffusion parfaite est assurée par la notion de "multipermutation".

On peut également étudier d'un point de vue formel la diffusion globale par les propriétés géométriques des réseaux de calcul (et l'on obtient des bornes inférieures pour la complexité des attaques génériques), ou par comptage du nombre minimal de boîtes actives dans une caractéristique différentielle ou linéaire.

La qualité de la confusion dépend des coefficients $\mathrm{DP}_{\max}$ et $\mathrm{LP}_{\max}$ des boîtes de calcul. On connaît les valeurs optimales de ces coefficients, mais il vaut mieux se contenter de s'en approcher, car les fonctions qui les atteignent recellent d'autres propriétés indésirables.

Ces critères simples permettent d'assurer une forme de sécurité heuristique. On peut cependant offrir une sécurité prouvée face à une classe d'attaques bien définie grâce à la notion de décorrélation. Ce concept rassemble en fait les idées de confidentialité parfaite de Shannon, les notions de fonctions universelles de Carter-Wegman, et la méthode de preuve de sécurité par "dérandomisation" de Luby-Rackoff. On peut ainsi démontrer la sécurité face aux attaques différentielles, linéaires, et plus généralement, itératives. On note que cette approche offre plus de flexibilité que la construction *ad hoc* de Nyberg-Knudsen.

Chapitre 4

Mise en Œuvre

> *"En chiffrement conventionnel, aucun mécanisme n'est enfoui au plus profond de l'algorithme, et c'est chaque pièce de l'assemblage qui doit être vérifiée : les techniques de cryptanalyse différentielle et linéaire forment une sorte de banc d'essai, qui permet de s'assurer de l'absence de défaut visible."*
> Jacques Stern
> In *La Science du Secret*, 1998.

On présente dans ce chapitre deux exemples de réalisation concrète de procédés de chiffrement symétrique à partir des théories développées dans le chapitre 3.

- CS-Cipher, qui offre une sécurité heuristique et d'excellentes performances,

- DFC, qui offre une sécurité prouvée par décorrélation pour des performances raisonnables.

La première fonction a été construite dans un but commercial pour permettre un chiffrement à haut débit. Elle est présentée en annexe G.[1] La seconde est un prototype qui a été soumis à la première phase de standardisation AES de la chambre de commerce du gouvernement américain.[2] Comme on en a déjà vu les grands principes de construction dans les chapitres précédents, on se contente ici de présenter leurs caractéristiques.

[1] Elle a été présentée au colloque *Fast Software Encryption* 98 [155].
[2] Voir [55, 56].

4.1 CS-Cipher

On présente ici brièvement les particularités du procédé de chiffrement CS-Cipher.[3]

4.1.1 Présentation de CS-Cipher

CS-Cipher est construit comme un réseau de substitutions-permutations au moyen de boîtes de calcul réversibles. On utilise en fait un réseau FFT à 2^3 entrées (chaque entrée étant un octet, on traite des blocs de message de 64 bits) avec des multipermutations. On utilise la construction du théorème 3.2 avec $n = 8$ et $c = \mathtt{55}_x$, et où σ est une rotation circulaire de un bit vers la gauche. On compose également chaque sortie avec une permutation P telle que $\mathrm{DP}_{\max}(P) = \mathrm{LP}_{\max} = 2^{-4}$. En fait, si x_l et x_r sont deux octets, on définit

$$M(x_l|x_r) = (P(\varphi(x_l) \oplus x_r), P(R_l(x_l) \oplus x_r))$$

où

$$\varphi(x_l) = (R_l(x_l) \text{ AND } \mathtt{55}_x) \oplus x_l$$

et où R_l est la rotation circulaire d'un bit vers la gauche. On note que le choix de la constante $\mathtt{55}_x$ procure à φ une propriété d'involution, ce qui montre qu'il est facile de définir M^{-1} (élément indispensable pour programmer le déchiffrement).

La permutation P est en fait une involution $P = \Psi(f, g, f)$ où f et g sont judicieusement choisies pour

- assurer à P les valeurs $\mathrm{DP}_{\max}$ et $\mathrm{LP}_{\max}$ voulues,
- être facilement réalisées par un circuit booléen.

Dans la plupart des implémentations, on représente P par une table. Comme P est une involution, une seule table suffit, ce qui est appréciable lorsque la place en mémoire est très limitée comme c'est le cas dans une carte à puce. On souhaite cependant rendre le câblage de CS-Cipher possible en technologie VLSI, ce qui impose d'utiliser des circuits booléens de faible taille.

A chaque étage du réseau FFT, on effectue une opération $\oplus$ avec une clef ou une constante, et quatre applications de M en parallèle (voir Fig. G.3 p. 190). On a ainsi 24 étages.

[3] On pourra se reporter à l'annexe G pour plus de détails.

4.1.2 Sécurité de CS-Cipher

Comme mentionné dans la section 3.1.3, on peut majorer les coefficients DP et LP de toute caractéristique de CS-Cipher par comptage du nombre minimal n de boîtes P actives, en élevant la valeur $\mathrm{DP}_{\max}(P)$ ou $\mathrm{LP}_{\max}(P)$ à la puissance n.

On considère que la multipermutation M est composée d'une multipermutation linéaire M_0 et de deux boîtes P en sortie. D'après la propriété de multipermutation et le critère que doit satisfaire une caractéristique sur une boîte linéaire (à savoir l'équation (2.1) pour une caractéristique différentielle, et l'équation (2.10) pour une caractéristique linéaire), pour chaque boîte M_0 active, il existe au moins trois des quatre arêtes adjacentes qui sont actives. En fait chaque arête qui relie deux boîtes M_0 "traverse" une boîte P.

Le problème revient donc à chercher un sous-ensemble non vide d'arêtes de taille minimale dans le graphe FFT à 2^8 entrées et 24 étages avec la règle que le nombre d'arêtes adjacentes à un même sommet quelconque dans ce sous-ensemble ne peut être égal qu'à 0, 3 ou 4. Pour le résoudre, à chaque étage i d'arêtes on associe un vecteur v_i de huit bits qui correspond à l'emplacement des arêtes du sous-ensemble. La suite $v_1, \ldots, v_{24}$ doit satisfaire des règles de transition $v_i v_{i+1}$ pour tout i. On peut constituer un graphe $G = (V, E)$ où V est l'ensemble des 256 vecteurs possibles (255 en fait, car on retire le vecteur nul qui est un état impossible) et E est l'ensemble des transitions possibles. A chaque sommet on associe son poids de Hamming (le nombre de bits à "1"), et le problème revient donc à chercher dans G un chemin de longueur 24 qui ne soit pas de poids minimal, ce qui se résout par des algorithmes standards.

On montre ainsi qu'un tel chemin a nécessairement un poids supérieur à 72, donc qu'une caractéristique a nécessairement 72 boîtes P actives. Une caractéristique a donc un coefficient DP ou LP inférieur à 2^{-288}, ce qui garantit une sécurité heuristique face aux attaques différentielles et linéaires.

On peut également effectuer cette analyse avec la notion de caractéristique différentielle tronquée.

4.1.3 Implémentations de CS-Cipher

L'algorithme CS-Cipher peut être facilement câblé dans une technologie VLSI. On estime à 1216 le nombre total de portes NAND nécessaires pour réaliser un étage complet (donc quatre fonctions M) dans un circuit de profondeur 26. On peut donc envisager un circuit fonctionnant à 30MHz sur $1\mathrm{mm}^2$ "dans le coin" d'un microprocesseur. On chiffre ainsi en 24 cycles, soit à une vitesse de 73Mbps. On peut faire un *pipeline* de ces circuits, et obtenir

un ASIC de 15mm^2 de 30000 portes NAND qui chiffrent à 2Gbps pour une vitesse d'horloge de 30MHz.

Comme CS-Cipher est orienté vers des architectures à 8 bits, il est très facile d'implémenter cet algorithme sur une carte à puce fonctionnant à 4MHz avec un microprocesseur 6805. Un programme (non optimisé) a permis d'obtenir une vitesse de 20Kbps (soit moins de 4ms par chiffrement de bloc, ce qui est déjà très rapide).

Sur des microprocesseurs modernes fondés sur une architecture à 32 bits, on obtient par un programme très simple écrit en langage C une vitesse de 2Mbps à 133MHz, ce qui est raisonnable. En assembleur sur Pentium, on peut atteindre 8Mbps (le programme n'étant pas encore optimisé).

De plus, on peut tirer avantage de la technique de programmation "*bit-slice*" de Biham.[4] Avec cette méthode, on programme l'algorithme comme si l'on avait un microprocesseur avec des registres de 1 bit, et l'on fait ainsi des opérations en parallèle comme sur une architecture SIMD. Avec un microprocesseur Alpha biprocesseur fonctionnant à 300MHz, on peut atteindre un débit de chiffrement de 196Mbps.

Les performances de CS-Cipher en font donc l'un des algorithmes les plus rapides.

4.2 DFC

On présente ici brièvement les particularités du procédé de chiffrement DFC.[5]

4.2.1 Présentation de DFC

DFC est un schéma de Feistel à huit étages pour des blocs de message de 128 bits. Chaque fonction d'étage est une fonction $\mathrm{RF}_{\mathrm{RK}_i}$ qui utilise une sous-clef $\mathrm{RK}_i = \mathrm{ARK}_i|\mathrm{BRK}_i$ et de la forme

$$\mathrm{RF}_{\mathrm{RK}_i}(x) = \mathrm{CP}(((\mathrm{ARK}_i \times x + \mathrm{BRK}_i) \bmod (2^{64} + 13)) \bmod 2^{64})$$

où CP est une permutation. En fait, on choisit CP pour perturber les structures linéaires utilisées dans cette construction : l'arithmétique modulo $2^{64} + 13$ et la linéarité au sens de $\oplus$. On utilise ainsi une addition modulo 2^{64} et un accès à une table.

[4] Dans un article présenté à *Fast Software Encryption* 97, Biham a montré que l'on pouvait implémenter DES avec cette technique pour atteindre un débit de 137Mbps à 300MHz sur un microprocesseur Alpha biprocesseur [26].

[5] On pourra se reporter à l'annexe H pour plus de détails.

Les huit sous-clefs RK_i sont calculées par un procédé de génération pseudo-aléatoire à partir d'une clef K de longueur arbitraire (256 bits au plus). On utilise en fait deux applications linéaires $\mathrm{EF}_1(K)$ et $\mathrm{EF}_2(K)$ pour transformer la clef en quatre sous-clefs, et la fonction de chiffrement réduite à quatre étages qui utilise alternativement EF_1 et EF_2 comme sous-clefs pour calculer par itérations successives les huit sous-clefs RK_i. Plus précisément, on définit $\mathrm{RK}_0 = 0$ et

$$\mathrm{RK}_{2i+1} = \mathrm{Enc}_{\mathrm{EF}_1(K)}(\mathrm{RK}_{2i}) \quad (4.1)$$
$$\mathrm{RK}_{2i+2} = \mathrm{Enc}_{\mathrm{EF}_2(K)}(\mathrm{RK}_{2i+1}) \quad (4.2)$$

où $\mathrm{Enc}_{\mathrm{EF}_i(K)}$ est la fonction de chiffrement réduite sur quatre étages qui utilise $\mathrm{EF}_i(K)$ comme chaîne de sous-clefs. Ce procédé coûte donc en nombre de calcul l'équivalent de quatre chiffrements de blocs.

4.2.2 Sécurité de DFC

Comme on utilise la construction décorrélée à l'ordre 2 de l'exemple 3.17 pour fonction d'étage, le théorème 3.21 assure $\mathrm{DecP}^2(\mathrm{DFC}) \leq 2^{-113.3}$. (Voir section 3.3.) Ceci permet d'estimer une borne inférieure de complexité pour des classes de distingueurs admettant une probabilité de succès d'au moins 10%.

- Un distingueur différentiel nécessite $n \geq 2^{110}$ (Th. 3.22 et 3.23).
- Un distingueur linéaire nécessite $n \geq 2^{92}$ (Th. 3.22 et 3.24).
- Un distingueur itératif d'ordre 1 à texte clair connu nécessite $n \geq 2^{48}$ (Th. 3.26).

On note que ces bornes s'appliquent à une fonction de chiffrement réduite à six étages au lieu de huit, ce qui procure à DFC une grande sécurité

Ces résultats sont toutefois conditionnés à l'hypothèse que les sous-clefs RK_i sont de distribution uniforme et indépendantes. On s'en affranchit par une hypothèse plus réaliste.

Hypothèse 4.1. *Pour $i = 1$ et 2, on ne peut pas construire un distingueur effectif qui admette un avantage supérieur à 1% entre la fonction de chiffrement réduite à quatre étages utilisant $\mathrm{EF}_i(K)$ (où K est de distribution uniforme) et une permutation aléatoire, et limité à seulement quatre requêtes et avec un budget de* \$1.000.000.000.

Ici, c'est le budget qui limite la puissance de la machine qui réalisera un distingueur. Une telle hypothèse permet de démontrer que les résultats de sécurité précédents s'appliquent à DFC lorsque le budget de l'attaquant est limité.

Par extrapolation de l'évolution des technologies, on démontre qu'il ne sera pas possible d'effectuer une recherche exhaustive sur 128 bits de clef dans le siècle à venir.

4.2.3 Clefs Faibles

Coppersmith a récemment montré qu'il existait des clefs particulièrement faibles pour DFC.

En effet, lorsque $\mathrm{RK}_2 = 0$ pour une clef K donnée, comme on a $\mathrm{RK}_0 = 0$, les équations (4.1) et (4.2) permettent de démontrer que l'on a $\mathrm{RK}_{2i} = 0$ pour tout i et une même valeur RK_{2i+1} indépendante de i. La conséquence sur le chiffrement est que la valeur $\mathrm{RF}_{\mathrm{RK}_{2i}}(x)$ ne dépend pas de x (comme $\mathrm{RK}_{2i} = 0$, on a $\mathrm{ARK}_{2i} = 0$, et la multiplication de x par 0 supprime la diffusion de l'information x). Le chiffrement revient donc à remplacer un étage sur deux du schéma de Feistel par un XOR avec une constante c. De plus, les étages restants utilisent la même sous-clef RK_{2i+1}, donc une même fonction f.

Si l'on écrit l'effet de deux étages successifs, on obtient

$$\mathrm{Enc}_{\mathrm{RK}_1|\mathrm{RK}_2}(x_l|x_r) = x_l \oplus f(x_r)|x_r \oplus c$$

pour une certaine fonction f et une certaine constante c. On peut alors écrire l'effet de quatre étages successifs : on a

$$\mathrm{Enc}_{\mathrm{RK}_1|\mathrm{RK}_2|\mathrm{RK}_3|\mathrm{RK}_4}(x_l|x_r) = x_l \oplus f(x_r) \oplus f(x_r \oplus c)|x_r.$$

Enfin, pour le chiffrement complet (huit étages), on a

$$\mathrm{DFC}_K(x_l|x_r) = x_r|x_l$$

ce qui est très facile à décrypter !

Les clefs faibles sont donc les clefs K telles que $\mathrm{RK}_2 = 0$. Une telle égalité apparaît avec une probabilité d'environ 2^{-128}.

Il convient donc de s'interroger sur la signification de telles clefs faibles. Tout d'abord, leur existence ne contredit en rien les résultats de sécurité, car la densité des clefs faibles est tellement petite que leur influence sur la complexité en moyenne des attaques est négligeable. Enfin, une densité de 2^{-128} signifie qu'il faudrait essayer environ 2^{128} clefs aléatoires avant d'en obtenir

une qui soit faible, ce qui est *a priori* aussi coûteux qu'une recherche exhaustive sur 128 bits. Il est donc peu probable que l'on trouve accidentellement une clef faible dans le siècle prochain.

La seule chose qu'il faut craindre, c'est qu'une telle clef faible soit malintentionnellement construite : un centre chargé de délivrer des clefs à des utilisateurs peut être corrompu pour qu'une clef faible soit fournie à l'un d'entre eux, l'empêchant ainsi de protéger ses communications. Une telle clef peut être construite par une attaque dans le milieu de complexité 2^{64}. Il est probable qu'une attaque dédiée puisse abaisser considérablement ce nombre.

Pour se protéger de telles attaques, il suffit évidemment de vérifier que la clef n'est pas faible avant de l'utiliser. Cela pose cependant le problème de la preuve de sécurité : est-on certain qu'il n'existe pas d'autres types de clefs faibles ? Des recherches futures permettront sans doute de construire d'autres procédés avec une assurance d'équivalence entre les clefs.

4.2.4 Implémentations de DFC

DFC se prête difficilement à une construction VLSI ou à une implémentation avec la technique du "*bit-slice*". Cela est dû à l'emploi de la multiplication sur des nombres de 64 bits.

On peut cependant implémenter DFC en langage machine sur la plupart des microprocesseurs actuels. Sur Pentium 200MHz (architecture sur 32 bits), on atteint un taux de chiffrement de 34Mbps, mais DFC est plus orienté vers les futurs microprocesseurs de 64 bits. Sur Alpha 600MHz (qui ressemble certainement plus aux microprocesseurs du futur proche), on atteint 138Mbps.

On peut également mettre DFC dans une carte à puce fondée sur un microprocesseur 6805. Un problème de limitation de la mémoire vive se pose, ce qui empêche, dans les cartes à puce les moins chères, de maintenir les sous-clefs RK_i en mémoire, mais oblige à les recalculer au vol. Si l'on parvient à enregistrer les RK_i, on peut chiffrer un bloc de message en 9ms à 4MHz (soit 14Kbps), ce qui est honorable. Dans le cas contraire, le calcul est cinq fois plus lent, soit 50ms (ou 3Kbps). Les cartes à puce du futur utiliseront des architectures de 32 bits, ce qui améliorera considérablement ces performances.

4.3 Conclusion

CS-Cipher est donc un système de chiffrement très performant et adapté au marché actuel. Il offre une grande sécurité (heuristique) face aux attaques

connues et réalise un bon compromis entre l'état de l'art et les coûts de mise en œuvre.

En revanche, DFC est une nouvelle architecture orientée vers la technologie de demain et elle offre une nouvelle notion de sécurité. On prouve ainsi que DFC est immunisé contre certaines classes d'attaques. Ce domaine est cependant encore ouvert à de nouvelles améliorations.

Conclusion

Alors que la recherche dans le domaine du chiffrement par blocs était auparavant un amalgame de propositions empiriques de nouveaux algorithmes et de nouvelles attaques dédiées, on constate à présent qu'un embryon de théorie se dégage.

Si l'on cherche, tout d'abord, à classer les propositions d'algorithmes, on dégage deux principales méthodes : le schéma de Feistel et ses généralisations, et les autres "réseaux de substitutions-permutations" construits à partir de fonctions réversibles. En fait, on constate que l'on peut décrire les algorithmes par une notion géométrique abstraite de "réseau de calcul" sur lequel sont branchées des "boîtes de calcul". Un tel formalisme permet directement de définir les notions de "diffusion" et de "confusion", ainsi que les méthodes d'analyse "différentielles" et "linéaires". Une fois définis des outils standards de manipulation des réseaux, des idées naturelles émergent, comme la notion de "projection de réseau" qui permet de généraliser les méthodes d'attaques évoquées et de les améliorer. Ainsi peut-on améliorer les attaques de DES.

Ce concept de réseau procure également un pouvoir d'abstraction qui permet d'envisager des "attaques génériques" et d'en évaluer la complexité. Pour cela, on oublie l'orientation du réseau et la description précise des boîtes de calcul pour obtenir la notion de "graphe d'équations". On peut alors généraliser les attaques génériques bien connues comme la "recherche exhaustive" et l'"attaque dans le milieu", ce qui permet de déceler les faiblesses géométriques intrinsèques des réseaux. On peut, par exemple, calculer le nombre d'étages minimal d'un réseau particulier pour que ces faiblesses s'estompent.

Si l'on cherche à protéger des procédés de chiffrement contre des méthodes d'attaques connues, on peut tout d'abord assurer une "sécurité heuristique". Ainsi peut-on renforcer la diffusion — par exemple au moyen de la notion de "multipermutation" — et la confusion — par exemple en approchant les bornes théoriques des coefficients $\mathrm{DP}_{\max}$ et $\mathrm{LP}_{\max}$. On peut ensuite s'assurer que les caractéristiques différentielles et linéaires ont toutes des coefficients DP ou LP petits. Le procédé de chiffrement CS-Cipher réalise un bon com-

promis entre cet état de l'art et le coût de mise en œuvre.

D'autres méthodes permettent de prouver une sécurité face à ces mêmes attaques. On mentionne l'approche *ad hoc* de Nyberg-Knudsen, utilisée notamment dans la fonction MISTY. Une autre méthode offre plus de flexibilité de construction : le principe de la "décorrélation". Cette notion effectue en fait le lien entre les divers résultats de sécurité connus : la sécurité de Shannon, la notion de fonction universelle de Carter-Wegman, et la sécurité de Luby-Rackoff. Elle fournit des outils pour démontrer la résistance face à des classes d'attaques : les méthodes différentielles, linéaires, et plus généralement "itératives". Le procédé de chiffrement DFC, dédié aux technologies de demain, est un prototype qui exploite au mieux ces outils.

Un tel cadre offre de nouvelles perspectives pour le chiffrement symétrique. On peut envisager, dans la continuité de ces travaux, de démontrer la sécurité face à d'autres classes d'attaques, d'améliorer les bornes obtenues jusqu'à présent, *etc.* Le problème d'établir une méthode plus systématique de calcul de la décorrélation se pose également : notamment, peut-on calculer la décorrélation d'une fonction décrite par un réseau de calcul ? Y a-t-il de meilleurs schémas que celui de Feistel ? Ces questions sont autant de sujets de futures recherches qui permettront de répondre aux besoins industriels de sécurité.

Cette présentation montre en fait qu'en vingt cinq ans de recherche publique, et grâce notamment, comme le montrent les références, aux organisateurs et aux participants du colloque *Fast Software Encryption* depuis 1993, le domaine du chiffrement symétrique est passé d'une "phase empirique" à une "phase heuristique". A présent, des outils formels de preuve sont disponibles, et une nouvelle phase peut commencer.

Annexe A

FFT-Hash-II is not yet Collision-free

[*Cet article de* Serge Vaudenay *a été publié dans les actes du colloque Crypto 92 [161].*]

Abstract. In this paper, we show that the FFT-Hash function proposed by Schnorr [143] is not collision free. Finding a collision requires about 2^{24} computation of the basic function of FFT. This can be done in few hours on a SUN4-workstation. In fact, it is at most as strong as a one-way hash function which returns a 48 bits length value. Thus, we can invert the proposed FFT hash-function with 2^{48} basic computations. Some simple improvements of the FFT hash function are also proposed to try to get rid of the weaknesses of FFT.

History

The first version of FFT-Hashing was proposed by Schnorr during the rump session of Crypto'91 [141]. This function has been shown not to be collision free at Eurocrypt'92 [20]. An improvement of the function has been proposed the same day [143] without the weaknesses discovered. However, FFT-Hashing has still some other weaknesses as it is proved in this paper.

A.1 FFT-Hash-II, Notations

The FFT-hash function is built on a basic function $< . >$ which takes one 128-bits long hash block H and one 128-bits long message block M, and return a 128-bits long hash block $< H, M >$. The hash value of n message

blocks $M_1, \ldots, M_n$ is $< \ldots << H_0, M_1 >, M_2 >, \ldots, M_n >$ where H_0 is a constant given in hexadecimal by :

$$H_0 = 0123\ 4567\ 89ab\ cdef\ fedc\ ba98\ 7654\ 3210$$

The basic function is defined by two one-to-one functions Rec and FT2 on the set $(\mathrm{GF}_p)^{16}$ where $p = 2^{16} + 1$. The concatenation HM defines 16 16-bits numbers which represents 16 numbers in GF_p between 0 and $p - 2$. $(\mathrm{Rec} \circ \mathrm{FT2} \circ \mathrm{Rec})(HM)$ defines 16 numbers of GF_p. The last 8 numbers taken modulo 2^{16} are the result $< H, M >$.

We define the following notations :

$$\begin{aligned} A(M) &= H_0 M \\ B(M) &= \mathrm{Rec}(A(M)) \\ C(M) &= \mathrm{FT2}(B(M)) \\ D(M) &= \mathrm{Rec}(C(M)) \end{aligned}$$

So, $< H_0, M >$ is the last 8 numbers of $D(M)$ taken modulo 2^{16}. We define X_i the i-th number of X (from 0 to 15), and $X[i, j]$ the list of the i-th to the j-th number of X.

If $x_i \in \mathrm{GF}_p$, $i = 0, \ldots, 15$, we define $y_{-3} = x_{13}$, $y_{-2} = x_{14}$, and $y_{-1} = x_{15}$. Then, following Schnorr :

$$y_i = x_i + y_{i-1}^* y_{i-2}^* + y_{i-3} + 2^i \tag{A.1}$$

where $y^* = 1$ if $y = 0$ and $y^* = y$ otherwise. Then, we let :

$$\mathrm{Rec}(x_0, \ldots, x_{15}) = y_0, \ldots, y_{15}$$

If $x_i \in \mathrm{GF}_p$, $i = 0, \ldots, 7$, we define :

$$y_j = \sum_{i=0}^{7} \omega^{ij} x_i$$

where $\omega = 2^4$. Then, we define $FT(x_0, \ldots, x_7) = y_0, \ldots, y_7$.

If $x_i \in \mathrm{GF}_p$, $i = 0, \ldots, 15$, we define $y_0, y_2, \ldots, y_{14} = FT(x_0, x_2, \ldots, x_{14})$ and $y_1, y_3, \ldots, y_{15} = FT(x_1, x_3, \ldots, x_{15})$. Then, we define $\mathrm{FT2}(x_0, \ldots, x_{15}) = y_0, \ldots, y_{15}$.

A.2 Basic Remarks

If we want to find a collision to the hash function, we may look for a pair (x, x') of two 128-bits strings such that $< H_0, x >=< H_0, x' >$. In fact, we will look for x and x' such that $D(x)[8, 15] = D(x')[8, 15]$.

First, we notice that we have necessarily $C(x)[11,15] = C(x')[11,15]$. In one direction, we show that $C(x)_i = C(x')_i$ for $i = 11, \ldots, 15$. This is due to the equation :

$$C_i = D_i - D^*_{i-1}D^*_{i-2} - D_{i-3} - 2^i$$

Conversely, if we have both $C(x)[11,15] = C(x')[11,15]$ and $D(x)[8,10] = D(x')[8,10]$, then we have $D(x)[8,15] = D(x')[8,15]$.

Moreover, we notice on the equation A.1 that $B(x)[0,7]$ is a function of $x[5,7]$ only. Let us denote :

$$B(x)[0,7] = g(x[5,7])$$

Finally, we notice that $FT2$ is a linear function.

A.3 Breaking FFT

A.3.1 Outlines

If we get a set of 3.2^{24} strings x such that $C(x)[11,15]$ is a particular string R chosen arbitrarily[1], we will have a collision on $D(x)[8,10]$ with probability 99% thanks to the birthday paradox. We will describe an algorithm which gives some x with the definitively chosen R for any $x[5,7] = abc$.

Given $abc = x[5,7]$, we can compute $B(x)[0,7] = g(abc)$. If we denote $y = B(x)[8,15]$, the following equation is a linear equation in y ;

$$FT2(g(abc)y)[11,15] = R \tag{A.2}$$

We can define a function ϕ_R and three vectors U_e, U_o, U'_e such that :

$$(A.2) \Longleftrightarrow \exists \lambda, \lambda', \mu \quad y = \phi_R(abc) + \lambda U_e + \lambda' U'_e + \mu U_o$$

(see section A.3.2).

Finally, the system :

$$\left\{ \begin{array}{rcl} x[5,7] & = & abc \\ C(x)[11,15] & = & R \end{array} \right.$$

is equivalent to the system :

$$\left\{ \begin{array}{rcl} x[5,7] & = & abc \\ y & = & \phi_R(abc) + \lambda U_e + \lambda' U'_e + \mu U_o \\ H_0 x & = & \mathrm{Rec}^{-1}(g(abc)y) \end{array} \right.$$

[1]For the collisions found in this paper, R is the image of my phone number by $FT2$.

Which is equivalent to :

$$\begin{cases} y &= \phi_R(abc) + \lambda U_e + \lambda' U'_e + \mu U_o \\ y_{13} &= a + y^*_{12} y^*_{11} + y_{10} + 2^{13} \\ y_{14} &= b + y^*_{13} y^*_{12} + y_{11} + 2^{14} \\ y_{15} &= c + y^*_{14} y^*_{13} + y_{12} + 2^{15} \\ x[5,7] &= abc \\ x[0,4] &= \mathrm{Rec}^{-1}(g(abc)y)[8,12] \end{cases} \tag{A.3}$$

Is we substitute y by the expression of the first equation in the other equations, we obtain a system of three equations of three unknown λ, λ', μ. This system can be shown linear in λ and λ' by a good choice of U_e, U_o and U'_e. Then, this system can have some solutions only if the determinant, which is a degree 2 polynomial in μ is 0. This can gives some μ. Then, the number of (λ, λ') is almost always unique. For more details, see section A.3.3.

Finally, this gives 0 or 2 solutions x, with an average number of 1 for a given *abc*. If we try $1 \leq a < p$, $1 \leq b \leq 768$ and $c = 2$, we have 3.2^{24} *abc*.

A.3.2 Solving $(A.2)$

The function $X \longmapsto FT2(X)[11,15]$ is linear, and has a kernel of dimension 3. If we define :

$$\begin{aligned} U &= (0,0,0,0,4081,256,1,61681) \\ U' &= (0,0,0,0,65521,4352,1,0) \end{aligned}$$

we notice that :

$$\begin{aligned} FT(U) &= (482,56863,8160,57887,7682,0,0,0) \\ FT(U') &= (4337,61202,65503,544,61170,3855,0,0) \end{aligned}$$

Let us introduce the following notation :

$$(x_0,\ldots,x_7) \times (y_0,\ldots,y_7) = (x_0,y_0,\ldots,x_7,y_7)$$

We have $FT2(X \times Y) = FT(X) \times FT(Y)$. Thus, we can can define :

$$\begin{aligned} U_e &= U \times 0 \\ U_o &= 0 \times U \\ U'_e &= U' \times 0 \end{aligned}$$

So, we have :

$$\begin{aligned} U_e &= (0,0,0,0,0,0,0,0,4081,0,256,0,1,0,61681,0) \\ U_o &= (0,0,0,0,0,0,0,0,0,4081,0,256,0,1,0,61681) \\ U'_e &= (0,0,0,0,0,0,0,0,65521,0,4352,0,1,0,0,0) \end{aligned}$$

These vectors are a base of the kernel of $X \longmapsto FT2(X)[11,15]$.

If M denotes the matrix of FT, we can write it using four 4×4 blocks :

$$M = \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix}$$

If x and y are two vectors of 4 elements, we have :

$$FT(xy)[4,7] = 0 \Longleftrightarrow y = -M_{22}^{-1} M_{21} x$$

Let us define :

$$N = -M_{22}^{-1} M_{21} = \begin{pmatrix} 65281 & 4335 & 289 & 61170 \\ 3823 & 8992 & 53012 & 65248 \\ 8447 & 61748 & 56545 & 4335 \\ 4369 & 57090 & 3823 & 256 \end{pmatrix}$$

Now, if x and y are two vectors of 8 elements, we have :

$$FT2(xy)[8,15] = 0 \Longleftrightarrow y = Nx^0 \times Nx^1$$

Where $x = x^0 \times x^1$. Let us define :

$$\phi_R(abc) = 0(Nx^0 \times Nx^1 + y^0)$$

where $g(abc) = x^0 \times x^1$ and $R = FT2(0y^0)[11,15]$ for an arbitrary y^0 (one's phone number for instance). Then, $\phi_R(abc)$ is a vector which begins by $g(abc)$, and such that $FT2(\phi_R(abc))$ ends by a constant vector R.

So, we have :

$$(A.2) \Longleftrightarrow \exists \lambda, \lambda', \mu \quad y = \phi_R(abc) + \lambda U_e + \lambda' U'_e + \mu U_o$$

A.3.3 Solving (A.3)

If we hope that no y_i $(i = 11, 12, 13, 14)$ is equal to 0 (we may ultimately test this condition, and forget the solutions y which do not pass this test, but this will be very rare), the system :

$$\begin{cases} y &= \phi_R(abc) + \lambda U_e + \lambda' U'_e + \mu U_o \\ y_{13} &= a + y_{12}^* y_{11}^* + y_{10} + 2^{13} \\ y_{14} &= b + y_{13}^* y_{12}^* + y_{11} + 2^{14} \\ y_{15} &= c + y_{14}^* y_{13}^* + y_{12} + 2^{15} \\ x[5,7] &= abc \\ x[0,4] &= \mathrm{Rec}^{-1}(g(abc)y)[8,12] \end{cases}$$

imply :

$$\begin{aligned} z_{13}+\mu &= a+(z_{12}+\lambda+\lambda')(z_{11}+256\mu)+z_{10}+256\lambda+4352\lambda'+2^{13} \\ z_{14}+61681\lambda &= b+(z_{13}+\mu)(z_{12}+\lambda+\lambda')+(z_{11}+256\mu)+2^{14} \\ z_{15}+61681\mu &= c+(z_{14}+61681\lambda)(z_{13}+\mu)+(z_{12}+\lambda+\lambda')+2^{15} \end{aligned}$$

where $z=\phi_R(abc)$. If we define :

$$\begin{aligned} a' &= a+z_{12}z_{11}+z_{10}+2^{13}-z_{13} \\ b' &= b+z_{13}z_{12}+z_{11}+2^{14}-z_{14} \\ c' &= c+z_{14}z_{13}+z_{12}+2^{15}-z_{15} \end{aligned}$$

we have :

$$\begin{pmatrix} z_{11}+256\mu+256 & z_{11}+256\mu+4352 & a'-(1-256z_{12})\mu \\ z_{13}+\mu-61681 & z_{13}+\mu & b'+(256+z_{12})\mu \\ 61681\,(z_{13}+\mu)+1 & 1 & c'-(61681-z_{14})\mu \end{pmatrix} \begin{pmatrix} \lambda \\ \lambda' \\ 1 \end{pmatrix} = 0$$

This is a linear system of unknown λ and λ'. If this system has an equation, which determinant has to be 0.

A.3.4 Discussion

This condition may be sufficient in most of the cases. The determinant should be a degree 3 polynomial. However, the coefficient of μ^3 is the determinant of the following matrix :

$$\begin{pmatrix} 256 & 256 & (1-256z_{12}) \\ 1 & 1 & -(256+z_{12}) \\ 61681 & 0 & (61681-z_{14}) \end{pmatrix}$$

which is 0 since the first line is 256 time the second.

The coefficient of μ^2 is 0 with probability $1/p$, this is rare. In this case, we have one solution if the equation has a degree one, and zero or p solutions in the other cases.

μ has to satisfy a degree 2 equation. If the discriminant is different from 0, it has a square root with probability 50%. So, we have two different μ or no solution with probability 50%, and a single solution with probability $1/p$.

For each μ, we are likely to have a uniq solution (λ,λ'). However, it is possible to have 0 or p solutions, but it is rare. So, for each solution (λ,λ',μ), we can compute y in the system $(A.3)$, then x. Finally, we have zero or two solutions x in almost all cases.

A.3.5 Reduction of the Function FFT

To sum up, we have a function f_R such that for a given abc :

$$f_R(abc) = \{D(x)[8,10]; x[5,7] = abc \wedge C(x)[11,15] = R\}$$

$f_R(abc)$ is a list of 0 or 2 $D(x)[8,10]$ for each x such that $x[5,7] = abc$ and $C(x)[11,15] = R$. The average of number of x is 1, so f_R is almost a function.

The function f_R is a kind of reduction of FFT since a collision for f_R gives a collision for FFT. We can use the birthday paradox with f_R to get some collision. The expected complexity is $O(2^{24})$.

We can invert FFT with f_R to. If we are looking for x such that

$$D(x)[8,15] = z$$

we can compute $R = \mathrm{Rec}^{-1}(z)[11,15]$ and look for abc such that $f_R(abc) = z[0,2]$. The complexity is 2^{48}. Then, we get the x required.

A.4 Collisions with the Birthday Paradox

If we suppose that f_R is like a real random function, the probability that a set $\{f_R(x_i)\}$ for k different x_i have k elements is next to :

$$e^{-\frac{k^2}{2n}}$$

where n is the cardinality of the image of f_R, when k is next to $\sqrt{n}$. So, with $n = 2^{48}$ and $k = 3.2^{24}$, the probability is 1%.

Two collisions have been found in 24 hours by a SUN4 workstation with $k = 3.2^{24}$ different x. With the choice :

$$R = 5726\ 17fc\ b115\ c5c0\ a631$$

We got :

$$\begin{aligned} FFT(17b3\ 2755\ 4e52\ b915\ 2218\ 1948\ 00a8\ 0002) &= \\ FFT(9c70\ 504e\ 834c\ b15c\ f404\ 94e2\ 02a7\ 0002) &= \\ 0851\ 393d\ 37c9\ 66e3\ d809\ d806\ 5e8c\ 05b8 & \end{aligned}$$

and :

$$\begin{aligned} FFT(8ccc\ 23a4\ 086d\ fbb9\ 85f4\ 70b2\ 029e\ 0002) &= \\ FFT(9d53\ 45ae\ 3286\ ada7\ 8c77\ 9877\ 02b4\ 0002) &= \\ 10e5\ 49f5\ 9df0\ d91b\ 0450\ afcc\ fba4\ 2063 & \end{aligned}$$

Conclusion

The main weakness of FFT-Hash-II are described in section A.2. First, the beginning of the computation depends on too few information of the input : $B(x)[0,7]$ is a function of $x[5,7]$. Second, the output allows to compute too much information of the computations in FFT : $D(x)[8,15]$ allows to compute $C(x)[11,15]$. The connection between $B(x)$ and $C(x)$ is linear, this makes our attack possible.

To get rid of the first weakness, we might mix H_0 and x in $A(x)$ before applying Rec. Similarly, the result of $< H_0, x >$ should be the set of $D(x)_{2i+1}$ instead of the right side.

Acknowledgment

I am happy to thank JEAN-MARC COUVEIGNES, ANTOINE JOUX, ADI SHAMIR and JACQUES STERN from the *Groupe de Recherche en Complexité et Cryptographie* for any advices. I owe a lot of time to JACQUES BEIGBEDER, RONAN KERYELL and all the *Service des Prestations Informatiques* for hardware and software advices. Finally, I should thank *France Telecom* to have given to me a phone number which hid so many collisions.

Annexe B

On the Weak Keys of Blowfish

[*Cet article de* Serge Vaudenay *a été publié dans les actes du colloque* Fast Software Encryption *96 [164].*]

Abstract. Blowfish is a sixteen-rounds Feistel cipher in which the F function is a part of the private key. In this paper, we show that the disclosure of F allows to perform a differential cryptanalysis which can recover all the rest of the key with 2^{48} chosen plaintexts against a number of rounds reduced to eight. Moreover, for some weak F function, this attack only needs 2^{23} chosen plaintexts against eight rounds, and 3×2^{51} chosen plaintexts against sixteen-rounds. When the F function is safely kept private, one can detect whether it is weak or not with a differential attack using 2^{22} plaintexts against eight rounds.

Blowfish was proposed by Schneier in the Cambridge Security Workshop [136]. It appears to be a very fast encryption function when we always use the same private key. It is based on the Feistel cipher [51]. Differential cryptanalysis [30] is known to be one of the most powerful attack on this kind of cipher. The design of Blowfish includes the new feature that the s-boxes are randomly generated from the private key. Hence, for some particular *weak keys*, a differential cryptanalysis may be successful. This paper shows the first analysis of Blowfish, as an answer to the Dr Dobb's Journal Blowfish Cryptanalysis Contest proposed in [137].

B.1 Blowfish

Blowfish encrypts a 64-bit plaintext into a 64-bit ciphertext using a variable key length [136]. The encryption proceeds with a suggested number of $t = 16$

rounds. In the following, we may consider a smaller number of rounds t. First, the key is expanded into a 4168-bytes key following a scheduling scheme which works as a pseudo-random generator. As this scheme is very complicated, one has to store definitely the expanded key. Thus, it is reasonable to consider the expanded key as the *real* key in the attack.

The expanded key consists of:

- $t + 2$ 32-bit constants $P_1, \ldots, P_{t+2}$;
- four arrays of 256 32-bit values which describe four s-boxes $S_1, \ldots, S_4$ with 8-bit inputs and 32-bit outputs.

The four s-boxes define a 32-bit to 32-bit function F by

$$F([abcd]) = ((S_1(a) + S_2(b)) \oplus S_3(c)) + S_4(d)$$

where $\oplus$ is the bit-wise xor and $+$ is the addition modulo 2^{32} and $[abcd]$ is the concatenated bit string of the four 8-bit strings a, b, c and d.

The plaintext $\mathcal{P} = (L_0, R_0)$ is divided into two 32-bit halves. Each round is defined recursively following the Feistel scheme by

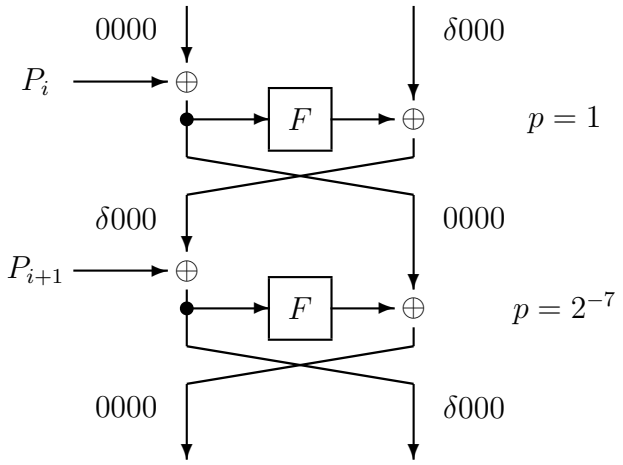
$$R_i = P_i \oplus L_{i-1} \text{ and } L_i = R_{i-1} \oplus F(R_i).$$

the ciphertext is $\mathcal{C} = (R_t \oplus P_{t+2}, L_t \oplus P_{t+1})$ (the left and right registers are not exchanged for the final round).

B.2 Known F - Weak Key Attack

All through this paper, the term *weak key* means there exists an s-box, say S_1, which has a collision. That is to say, there exist two different bytes a and a' such that $S_1(a) = S_1(a')$. In this section, we assume the opponent knows the part of the private key which describes the F function, that is the four s-boxes. (in fact, we only need to know a and a' to recover seven bits of information on the private key.)

Let δ denote the xor-difference of the collision of S_1 (that is $\delta = a \oplus a'$ with the previous notation) We consider the following iterative characteristic.



(Throughout this paper, figures represent 8-bit values so that $[\delta 000]$ is a 32-bit value.) Assuming there is only one collision for S_1 with difference δ, the probability of this characteristic is 2^{-7}.

For Blowfish reduced to $t = 8$ rounds, we iterate this characteristic three times as shown on figure B.1 ($xyzt$ represents an undetermined value). The resulting characteristic has probability 2^{-21}. Hence, trying 2^{21} chosen plaintext pairs with xor $[0000\delta 000]$, we easily detect a ciphertext pair $(\mathcal{C}, \mathcal{C}')$ with xor $[\delta 000xyzt]$. Note that for a random pair, the probability of getting such an xor is 2^{-32}. So, a detected pair with the good xor is certainly a good pair. With such a pair, let denote $\mathcal{C} = (L, R)$. Since we have

$$F(L \oplus P_{10}) \oplus F(L \oplus P_{10} \oplus [\delta 000]) = [xyzt]$$

we can try exhautively all the 2^{32} possible P_{10} until this equation holds. It is easy to check that Blowfish with t rounds and a known P_{t+2} is equivalent to Blowfish with $t - 1$ rounds. So, this attack allows to reduce the cipher to $t = 7$ rounds.

More generally, for Blowfish with t rounds, the same iterated characteristic has probability $2^{-7\times\lceil\frac{t-2}{2}\rceil}$. So, we can use $2^{7\times\lceil\frac{t-2}{2}\rceil}$ chosen plaintext pairs, and, for each ciphertext pair with xor $[\delta 000xyzt]$, list the possible values of P_{t+2}. A random pair has this xor with probability 2^{-32}, and each such pair suggests one value of P_{10} on average.

For $t \leq 10$, all pairs which have this xor are good, but for $t \geq 11$, we may get $2^{7\times\lceil\frac{t-2}{2}\rceil-32}$ wrong pairs. Each wrong pair suggests one random value for P_{t+2} on average. So, trying $3 \times 2^{7\times\lceil\frac{t-2}{2}\rceil}$ chosen plaintext pairs, we get three good pairs which suggest the same value with high probability, and no other value may be suggested more than two times for $t \leq 16$.

As the complexity of the attack on $t-1$ rounds is the same as for t rounds

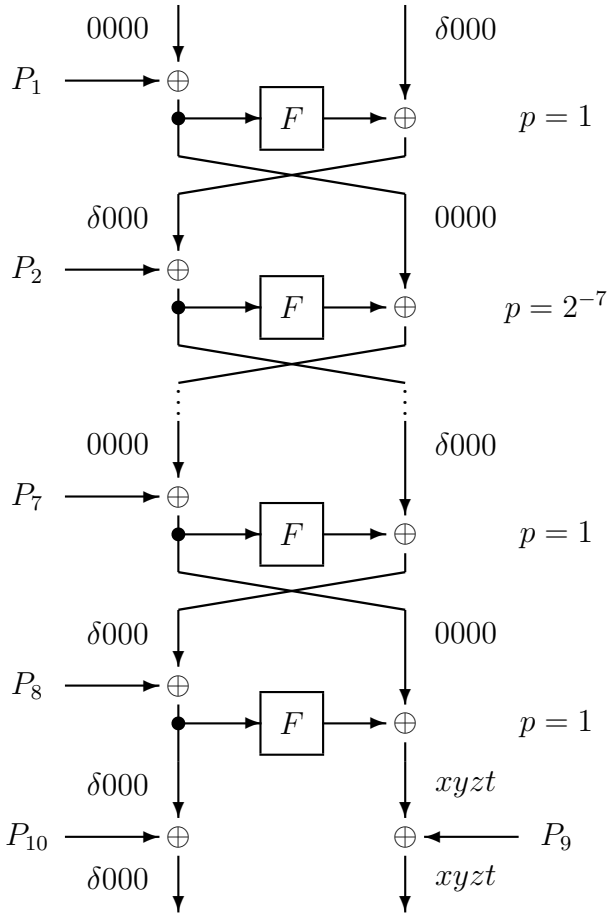


FIGURE B.1 – Characteristic for 8 rounds

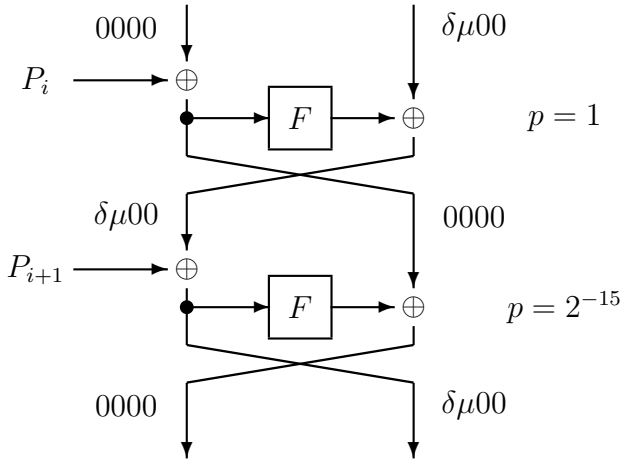
if t is even, the number of chosen plaintexts required is $3 \times 2^{2+7\times\lceil\frac{t-2}{2}\rceil}$. For $t = 16$, this is 3×2^{51}.

For $t = 8$, since there is no problem with wrong pairs, the number of plaintexts required is 2^{23}.

B.3 Known F - Random Key Attack

As in the previous section, we assume the description of the F function has been disclosed, but the private key is not necessarily supposed to be weak. The mapping $(a, b) \mapsto S_1(a) + S_2(b)$ is a 16 to 32 bits function, so it may have a collision $S_1(a) + S_2(b) = S_1(a') + S_2(b')$ with high probability.

Letting $\delta = a \oplus a'$ and $\mu = b \oplus b'$, we consider the following iterative characteristic.



Assuming there is only one collision for $S_1 + S_2$ with difference $\delta\mu$, the probability of this characteristic is 2^{-15}. Hence, for Blowfish with $t = 8$ rounds, this characteristic iterated as on figure B.1 has probability 2^{-45}, and 2^{46} chosen plaintext pairs include two good pairs and 2^{14} wrong pairs. So, the good value of P_{10} may be the only value suggested twice.

The attack on $t = 7$ rounds has the same complexity, so the number of plaintexts required is 2^{48}.

B.4 Weak Key Detection

What can we do without the description of F? We can try to detect whether a key is weak or not. For a random s-box S_1, the probability that there is no

collision is

$$\prod_{i=0}^{2^8-1}\left(1-\frac{i}{2^{32}}\right)=\frac{2^{32}!}{2^{32\times 2^8}(2^{32}-2^8)!}\approx 1-2^{-17.0}.$$

So, for a F function made with random s-boxes, the probability there exists a collision for one s-box is $2^{-15.0}$. Hence, one key out of 2^{15} may be weak. Now let us see how to distinguish which is weak by a chosen plaintext attack.

First one can bet on S_1 and use the characteristic on figure B.1. If we pick the bytes B_1, B_2, B_3, B_4, B_6, B_7 and B_8 at random (no B_5 here), in the *structure* of all the 2^8 plaintexts

$$\mathcal{P}=[B_1B_2B_3B_4B_5B_6B_7B_8]$$

there are 2^7 pairs with the good xor. Let

$$\mathcal{C}=[C_1C_2C_3C_4C_5C_6C_7C_8]$$

be the corresponding ciphertexts. In a good pair, we notice that

$$\mathcal{X}=[(B_5\oplus C_5)C_6C_7C_8]$$

must be the same for both messages of the pair. Thus, we can seek for pairs in the structure which makes $\mathcal{X}$ collide. If there are no good pairs, this occurs with probability roughly $2^{-17.0}$. Trying 2^{14} structures, we get one good pair with high probability, and no wrong pair with probability roughly 2^{-3}. Hence, with 2^{22} chosen plaintexts, we can detect a collision on S_1 and get the xor δ of the collision. The same attack holds for S_2, S_3 and S_4.

B.5 Conclusion

We have shown differential cryptanalysis on Blowfish is possible either against a reduced number of rounds or with the piece of information which describes the F function. This second case appears to be equivalent to an analysis done by Lee, Heys and Tavares [83] against the CAST cipher [11, 12]. In the analysis of CAST, the s-boxes are well design to resist to any attack while they are randomly generated in Blowfish. Compared to CAST, some of the s-boxes generated by Blowfish may be really weak, but it is not sure whether it is sufficient to mount an attack since they are supposed to be private.

We studied weaknesses of the s-boxes based on collisions. This way, we proved there are weak keys in Blowfish that enable to decrease significantly the complexity of the attacks (from 2^{48} to 2^{23} on eight rounds when F is known). We also showed it is possible to detect weak keys using 2^{22} chosen plaintexts (on eight rounds).

Acknowledgments

Many thanks to Lars Knudsen for his help and fruitful discussions.

Annexe C

An Experiment on DES: Statistical Cryptanalysis

[*Cet article de* Serge Vaudenay *a été publié dans les actes du colloque ACM Computer and Communications Security 96 [165].*]

Abstract. Linear cryptanalysis and differential cryptanalysis are the most important methods of attack against block ciphers. Their efficiency have been demonstrated against several ciphers, including the Data Encryption Standard. We prove that both of them can be considered, improved and joined in a more general statistical framework. We also show that the very same results as those obtained in the case of DES can be found without any linear analysis and we slightly improve them into an attack with theoretical complexity $2^{42.9}$.

We can apply another statistical attack — the χ^2-cryptanalysis — on the same characteristics without a definite idea of what happens in the encryption process. It appears to be roughly as efficient as both differential and linear cryptanalysis. We propose a new heuristic method to find good characteristics. It has found an attack against DES absolutely equivalent to Matsui's one by following a distinct path.

C.1 Introduction

Since the proposal of the Data Encryption Standard by the U.S. government, the scientific community concentrated a significant part of its efforts on its cryptanalysis [2]. This well-known function encrypts a 64-bits plaintext into a 64-bits ciphertext using a 56-bits secret key, so that the best attack is expected to have complexity 2^{56} (2^{55} if we take into account the complementation property of DES as in [62]).

A first significant result, obtained by Biham and Shamir, gave a general method for chosen plaintext attacks — the *differential cryptanalysis* [29, 30]. Using a deep analysis of the internal framework of the function, they try to control a correlated piece of information on several particular plaintexts and recover it by statistical attacks. The correlated piece of information used is simply a chosen bit-wise exclusive *or* difference between two texts. The main result of Biham and Shamir proves, using heuristic arguments, that it is possible to mount an attack with 2^{47} chosen plaintexts.

A second result gave also a general method, called *linear cryptanalysis*, for known plaintext attacks. It has been discovered by Matsui who proved that it is possible to implement an attack against DES with 2^{43} known plaintexts [91]. Using another deep analysis of the function, this attack tries to trace a correlation between one bit of information on the plaintext and one bit of information on the ciphertext. One more time, the information is obtained linearly with respect to the exclusive *or*.

Both methods are bottom-up approachs based on the concept of *characteristic*. This is a scenario of the propagation of the correlated piece of information. It is associated to a probability, which has to be as biased as possible. The goal of the heuristic arguments consists in finding *efficient* characteristics, first analyzing the linear properties of the substitution boxes, then plugging them into one another in a such a way that a linear information is leaked throughout the encryption process. Once this analysis has led to an efficient characteristic, we only need to keep which information on the plaintext and the ciphertext is required for the upper level of the attack.

The success of those methods have focused the attention on the linear properties of the boxes. In this paper, we try to prove that the linear properties are not so important. We propose another heuristic approach based on statistics. We show how to recover an attack similar to Matsui's one without any linear consideration. We also propose a top-down approach which unifies linear and differential cryptanalysis. We prove that a simple χ^2 test can get the similar results without knowing precisely what happens (for instance on a black box which implements a secret encryption function). Those results have already partially been presented in [163].

Throughout this paper, we use the following notations:

- k is a secret key in the domain $\mathcal{K}$;
- P is a plaintext in the domain $\mathcal{P}$;
- C is a ciphertext in the domain $\mathcal{C}$;

- Enc_k is an encryption function which maps P to C using key k;
- $x \wedge y$ denotes the bitwise *and* of the bit-strings x and y;
- $W(x)$ is the Hamming weight of the bit-string x;
- $x \cdot y$ is the dot-product of x and y, that is the parity of $W(x \wedge y)$;
- $1_{predicate}$ is 1 if *predicate* is true, 0 otherwise.

C.2 Heuristic using Projection

C.2.1 Transition Matrix of a Projected Cipher

In the encryption process, intermediate results of the encryption function can be arbitrarily ignored and supposed to be uniformly independent of the rest of the computation. We call this operation *projection*. After projection, assuming that the removed inputs are random, each box becomes stochastic, transforming the leftover inputs into the remaining outputs. Thus, it is possible to compute the *transition matrix* of the projected boxes.

For instance, if a and b are the masks of all remaining inputs and outputs of an S-box S (that is to say, that we only know the value $x \wedge a$ from the input x, and that we are only interested in the value $y \wedge b$ from the output y), we compute the matrix of all

$$T_{i,j} = \Pr_{X \text{ uniform}} \left[S(X) \wedge b = j / X \wedge a = i \right].$$

We use tools from tensorial algebra to compute the transition matrix of a network of S-boxes: the transition matrix of $(x, y) \mapsto (F(x), G(x))$ is the tensorial product (also called the Kronecker product) of the transition matrix of F and the transition matrix of G, and the transition matrix of $F \circ G$ is the matrix-product of the transition matrices of F and G whenever the output mask of G is the input mask of F. Assuming that M^i and M^o are the masks of the remaining inputs and outputs of the whole encryption function, we obtain all values

$$v_k^x = \Pr_{P \in \mathcal{P}, \mathcal{C} = \mathrm{Enc}_\parallel(\mathcal{P})} \left[(P \wedge M^i, C \wedge M^o) = x \right] - \frac{1}{q}$$

for all q possible values of x under the heuristic assumption, depending on the key k. This forms the *bias vector* $V_k = (v_k^{x_1}, v_k^{x_2}, \ldots)$ of the distribution

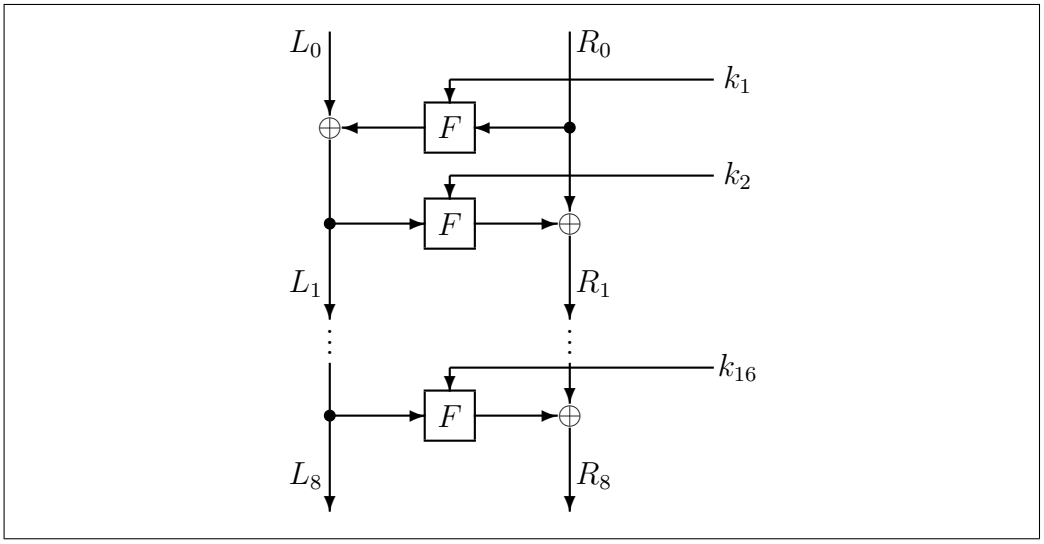


FIGURE C.1 – Feistel's scheme with 16 rounds

of $X = (P \wedge M^i, C \wedge M^o)$ with respect to the uniform distribution. In the following sections, we consider a more general X of the form

$$X = h_3(h_1(k), h_2(P, C))$$

depending on a small piece of information $h_1(k)$ on k and a small piece of information $h_2(P, C)$ on the pair (P, C).

C.2.2 Projection of DES

DES is based on the Feistel scheme illustrated on Figure C.1. There are two 32-bits registers L and R modified at each round by a process which depends on a subkey k_i depending on the master key k (see [2]).

As an example, we ignore the same 27 bits of the left register of DES and the same 31 bits of the right one between the second and the fifteenth round (the reason why we use only the 14 middle rounds will appear in the next Sections). More precisely, for all masks m_L and m_R such that $W(m_L) = 5$ and $W(m_R) = 1$, we only keep the information $(L_i \wedge m_L, R_i \wedge m_R)$ in each round, that is 6 bits of information. We computed the $2^6 \times 2^6$ bias vector $V_k(m_L, m_R)$ of

$$X = (L_1 R_0 \wedge m_L m_R, L_8 R_7 \wedge m_L m_R).$$

Experiments shows that the norm $||V_k(m_L, m_R)||_2$ which we call *deviation* does not depend significantly on k provided it is large. Thus, trying all the

possible positions of the 6 bits kept, we have found the best choices of the bit positions the first of are:

m_L	m_R	$\log_2 \|\|V_k\|\|_2$
$\mathtt{21040081}_{16}$	$\mathtt{00008000}_{16}$	-25.580
$\mathtt{21040082}_{16}$	$\mathtt{00008000}_{16}$	-25.583
$\mathtt{21040084}_{16}$	$\mathtt{00008000}_{16}$	-25.583
$\mathtt{21040088}_{16}$	$\mathtt{00008000}_{16}$	-25.583
$\mathtt{21040090}_{16}$	$\mathtt{00008000}_{16}$	-25.583
$\mathtt{210400a0}_{16}$	$\mathtt{00008000}_{16}$	-25.583
...	...	...

This shows that the best choices are exactly those which contain the pattern

$$m_L = \mathtt{21040080}_{16} \quad m_R = \mathtt{00008000}_{16}$$

(for which $\log_2 ||V_k||_2 = -24.583$) that is the bits used in Matsui's attack [91]. Trying all the 4 and 2 positions achieved an analogous result. Trying all the 3 and 3 positions did not provide any larger deviation.

The best other choice which does not contain Matsui's characteristic is:

$$m_L = \mathtt{04010104}_{16} \quad m_R = \mathtt{00c00000}_{16}$$

for which $\log_2 ||V_k||_2 = -30.768$.

C.2.3 Information on the Key Leaked

To see how much $V_k(m_L, m_R)$ depends on k, we studied how many different vectors we get with different k. Linear cryptanalysis only consider a one-bit long value X. Thus, the bias vector V_k has the form $(-\delta, \delta)$ and there are only two different vectors $(\mp\delta, \pm\delta)$, depending on one bit of information on k. Moreover, keys which produce the same bit of information are in the same affine space with codimension 1.

More generally, when a characteristic defined by (m_L, m_R) contains c linear characteristics, the vector $V_k(m_L, m_R)$ depends on c bits of information on k. Thus, there are 2^c different vectors, and keys which produce the same one are in the same affine space with codimension c. To study the nature of the information on k which influences the vector, we compute all the affine spaces spanned by random keys which produce the same vector. For instance, with the characteristic defined by

$$m_L = \mathtt{21040080}_{16} \quad m_R = \mathtt{00008000}_{16}$$

for which $\log_2 ||V_k||_2 = -24.583$ the experiment shows 4 different vectors $V_k(m_L, m_R)$. Thus, the key space is partitioned into 4 classes, and we can prove that each class spans an affine space with codimension 2. We already know that Matsui's linear characteristic defines one bit of information on k which is computed linearly:

$$\text{Parity}\left(\bigoplus_{i\in\{3,5,7,9,11,13,15\}} k_i \wedge 0000000020000000_8 \quad \oplus \right.$$
$$\left. \bigoplus_{i\in\{4,8,12\}} k_i \wedge 0400000000000000_8\right).$$

We have found another bit of information which influences the vector:

$$\text{Parity}\left(\bigoplus_{i\in\{2,4,6,8,10,12,14\}} k_i \wedge 0400000000000000_8\right).$$

Using Matsui's notations, those bits are respectively

$$k_3[22] \oplus k_4[44] \oplus k_5[22] \oplus k_7[22] \oplus k_8[44] \oplus k_9[22]$$
$$\oplus k_{11}[22] \oplus k_{12}[44] \oplus k_{13}[22] \oplus k_{15}[22]$$

and

$$k_2[44] \oplus k_4[44] \oplus k_6[44] \oplus k_8[44] \oplus k_{10}[44] \oplus k_{12}[44] \oplus k_{14}[44]$$

With the characteristic defined by

$$m_L = 04010104_{16} \quad m_R = 00c00000_{16}$$

we observed 16 different classes. We observed that keys in the same class spanned an affine space with codimension 4. Thus, this characteristic uses 4 linear bits on k.

C.3 Statistical Cryptanalysis

C.3.1 Model of the attack

In the model of the attack[1], the concept of *characteristic* defines three hash functions:

- $h_1 : \mathcal{K} \to \mathcal{L}$ where $\mathcal{L}$ is a small space with cardinality ℓ (the aim of the cryptanalysis is to obtain probabilistic information on $k' = h_1(k)$);

[1] This model appears to be similar to Harpes's partitioning cryptanalysis [59].

- $h_2 : \mathcal{P} \times \mathcal{C} \to \mathcal{S}$ where $\mathcal{S}$ is the *sample space* with cardinality s which only contains useful information for the analysis;
- $h_3 : \mathcal{L} \times \mathcal{S} \to \mathcal{Q}$ where $\mathcal{Q}$ is a space with cardinality q.

For a random sample $S = h_2(P, C)$ coming from random P and $C = \mathrm{Enc}_k(P)$, we let $X = h_3(k', S)$, $k' = h_1(k)$. Basically, X is a piece of information depending on the intermediate results in the encryption. For the purpose of the cryptanalysis, X should be both

- computable with small pieces of information on (P, C) and k, namely S and k',
- and sufficiently biased for $X = h_2(k', S)$ (where $k' = h_1(k)$) to be statistically distinguishable from the distribution of $h_2(K, S)$ comming from a wrong guess $K \neq k'$.

The principle of the attack consists in seeking for the good k' which makes the distribution of all the observed X deviate significantly from a smooth distribution. In the example of DES, h_3 is a mask over messages obtained after the first round and before the last round, and h_1 and h_2 give the information required to compute it.

We assume that we can use several independent samples $S = h_2(P, C)$, given that P follows a given distribution H in the domain $\mathcal{P}$ and such that $C = \mathrm{Enc}_k(P)$ with the unknown k. The attack is a known or chosen plaintext attack depending on whether H corresponds to an available real plaintext distribution or not. It may be a ciphertext only attack when S is computable from C. For all candidates K to $k' = h_1(k)$, we can compute a candidate $X = h_3(K, S)$ to $h_3(k', S)$. The main idea of the attack consists in assuming that we can distinguish $K = k'$ from $K \neq k'$ by a statistical measurement Σ on the observed distribution. In most cases, for $K = k'$, this distribution will look *less* regular than for $K \neq k'$. The attack proceeds in four phases:

- **Counting Phase.** Collect several random samples $S_i = h_2(P_i, C_i)$, $i = 1, \ldots, n$. This consists in counting all occurrences of all the possible values of S in s counters.
- **Analysis Phase.** For each of the ℓ candidates K, count all the occurrences in all $X_i = h_3(K, S_i)$ and give it a mark M using the statistic $\Sigma(X_1, \ldots, X_n)$. Hereafter n_x denotes (for a given K) the number of samples such that $h_3(K, S_i) = x$.
- **Sorting Phase.** Sort all the candidates K using their marks M_K.

- **Searching Phase.** Exhaustively try all keys following the sorted list of all the candidates.

The space complexity is $O(s+\ell)$ since we need s counters for all $S = h_2(P, C)$ and ℓ registers for all candidates K. The time complexity is $O(n)$ for the Counting Phase, $O(s\ell)$ for the Analysis Phase and $O(\ell \log \ell)$ for the Sorting Phase. The average complexity of the Searching Phase, which depends on the expected rank of the good candidate in the sorted list, will be discussed below. Typically, the bottleneck computations are the Counting Phase and the Searching Phase, and we need to study the trade-off between them: we need many samples to expect the good candidate to have a high rank, but not too many to be able to count them.

C.3.2 Analysis of the Attack

We make several approximations which might be justified by heuristic arguments in concrete examples. We recall that H denotes the distribution of the random plaintext source.

Approximation C.1. *If $K \neq h_1(k)$, the distribution $h_3(K, H)$ of*

$$X = h_3(K, h_2(P, \mathrm{Enc}_k(P)))$$

is a distribution D which does not depend on K.

Approximation C.2. *If $K = h_1(k)$, the distribution of X is a distribution D' which is independent on D.*

Typically, D is the uniform distribution in the domain $\mathcal{Q}$ with cardinality q. We call *deviation* between D and D' the value

$$d_2(D, D') = \sqrt{\sum_x \left(\Pr_{X \in D'}[X = x] - \Pr_{X \in D}[X = x] \right)^2}.$$

The accurate analysis depends on the choice of the statistic Σ, but we give here the outline of the analysis. We denote μ and σ (resp. μ' and σ') the mean and the standard deviation of $\Sigma(X_1, \ldots, X_n)$ all X_i following the distribution D (resp. D'). In the rest of this paper, we make another Approximation.

Approximation C.3. *We have $\sigma \approx \sigma'$ and $\mu \not\approx \mu'$.*

The mark M_K of K is defined to be

$$M_K = \frac{\Sigma(h_3(K, S_1), \ldots, h_3(K, S_n)) - \mu}{\sigma}$$

ϵ	0	2^{-2}	2^{-1}	1	2	2^2	2^3
$\Phi\left(-\epsilon/\sqrt{2}\right)$	50%	43%	36%	24%	$8\% = 2^{-3.7}$	$2^{-8.7}$	$2^{-27.0}$

Figure C.2 – Decreasing of the normal law

so, the standard deviation of any mark is 1, the expected mark of a wrong candidate is 0, and the expected mark of the good candidate is $\epsilon = \frac{\mu' - \mu}{\sigma}$ which will be called the *efficiency* of the attack. In real applications, Approximation C.3 may corresponds to a first order approximation.

Let

$$\Phi(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{t} e^{-\frac{t^2}{2}} dt.$$

be the normal distribution function. In the following, the sentence "*the distribution of M is asymptotically normal*" means that

$$\Pr\left[\frac{M - E(M)}{\sigma(M)} < t\right] \to \Phi(t)$$

when the number of samples is large.

Theorem C.4. *Under the Approximations and if the distribution of all M_K are asymptotically normal, the average complexity of the Searching Phase tends to*

$$\frac{N}{\ell} + \left(N - \frac{N}{\ell}\right) \Phi\left(-\epsilon/\sqrt{2}\right)$$

where N is the number of keys k.

This will be practically approximated by $N.\Phi\left(-\epsilon/\sqrt{2}\right)$.

Proof. The mark of the good candidate $k' = h_1(k)$ with samples $S_1, \ldots, S_n$ is

$$M_{k'} = \frac{\Sigma(h_3(k', S_1), \ldots, h_3(k', S_n)) - \mu}{\sigma}$$

which is approximately normal, with mean ϵ and standard deviation 1. The mark of any wrong candidate $K \neq h_1(k)$ is

$$M_K = \frac{\Sigma(h_3(K, S_1), \ldots, h_3(K, S_n)) - \mu}{\sigma}$$

which is approximately standardized and normal. $M_{k'}$ and M_K are independent by Approximation C.2, so $\frac{M_{k'} - M_K - \epsilon}{\sqrt{2}}$ is standardized and normal. Thus,

the probability that the $M_{k'}$ is less than M_K is $\Phi\left(-\epsilon/\sqrt{2}\right)$. The rank of k' is

$$1 + \sum_K 1_{M_{k'} < M_K}$$

so the expected rank of k' is $1+(\ell-1).\Phi\left(-\epsilon/\sqrt{2}\right)$ and the average complexity of the Searching Phase is obtained multiplying this by $\frac{N}{\ell}$. □

The decreasing of $\Phi\left(-\epsilon/\sqrt{2}\right)$ is illustrated on the table on Figure C.2.

C.3.3 The Use of Several Characteristics

It is possible to use several characteristics C_i (or the same one several times) with efficiency ϵ_i for $i = 1, \ldots, c$ using a trick analog to the one analyzed by Kaliski and Robshaw [71]. We get lists of several candidates so that each full key K have marks M_K^i. We let

$$\bar{\epsilon} = \sqrt{\sum_{i=1}^{c} {\epsilon_i}^2} \tag{C.1}$$

and we define the general mark

$$M_K = \sum_{i=1}^{c} \frac{\epsilon_i}{\bar{\epsilon}} M_K^i.$$

We can do the exhaustive search following the general marks. It is easy to prove that the Theorem C.4 remains valid if we replace ϵ by $\bar{\epsilon}$ when all M_K^i are independent. Thus, it is possible to slightly improve the best known linear attack on DES collecting a huge number of less efficient characteristics.

C.4 Differential Approach

For a given nonzero a, the statistic Σ_{diff} counts the number of sample pairs (X_i, X_j) such that $X_i \oplus X_j = a$:

$$\Sigma_{\text{diff}}(X_1, \ldots, X_n) = \sum_{i,j=1}^{n} 1_{X_i \oplus X_j = a} = \sum_{x \oplus y = a} n_x n_y.$$

For vectors a coming from a *differential characteristic*, a heuristic analysis from Biham and Shamir enables to approximate (for $i \neq j$)

$$\Pr_{X_i, X_j \in D'}[X_i \oplus X_j = a] - \Pr_{X_i, X_j \in D}[X_i \oplus X_j = a] = \delta.$$

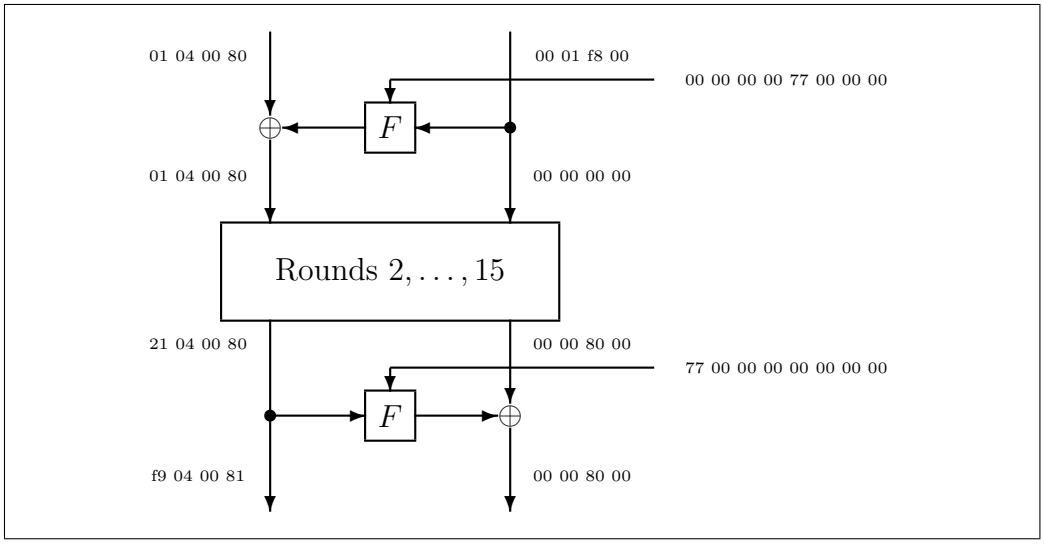


Figure C.3 – Matsui's characteristic

Theorem C.5. *If D is uniform over $\mathcal{Q}$, the efficiency of the attack using Σ_{diff} is*

$$\epsilon \approx n\frac{q}{\sqrt{2(q-1)}}\delta \leq n\frac{q}{\sqrt{2(q-1)}}(d_2(D,D'))^2.$$

Proof. We have $\mu' - \mu = n(n-1)\delta$ and

$$\sigma = \frac{\sqrt{n(n-1)}\sqrt{2(q-1)}}{q}$$

so we get ϵ. Using Cauchy-Schwarz's Inequality, we have

$$|\delta| = \left|\sum_{x\oplus y=a} v_k^x v_k^y\right| \leq (d_2(D,D'))^2$$

where v_k^x is defined in Section C.2. □

C.5 Linear Approach

C.5.1 Linear Cryptanalysis

For a given nonzero a, the statistic Σ_{lin} counts the number of samples X_i such that the dot product $X_i \cdot a$ is zero:

$$\Sigma_{\text{lin}} = \sum_{i=1}^{n} 1_{X_i\cdot a=0} = \sum_{x\cdot a=0} n_x.$$

For vectors a coming from a *linear characteristic*, a heuristic analysis from Matsui enables to approximate

$$\Pr_{X_i \in D'}[X_i \cdot a = 0] - \Pr_{X_i \in D}[X_i \cdot a = 0] = \delta.$$

Theorem C.6. *If D' is uniform over $\mathcal{Q}$, the attack using Σ_{lin}, which is asymptotically normal, is*

$$\epsilon = \sqrt{n}.2\delta \leq \sqrt{nq}.d_2(D, D').$$

This Theorem will be proved in a more general form below.

C.5.2 Matsui's Attack against DES

As illustrated by Figure C.3, Matsui's characteristic is defined by

$$\begin{aligned} k' &= \begin{pmatrix} k_1 & \wedge & \texttt{0000000077000000}_8 \\ k_{16} & \wedge & \texttt{7700000000000000}_8 \end{pmatrix} \\ S &= \begin{pmatrix} L_0R_0 & \wedge & \texttt{010400800001f800}_{16} \\ L_8R_8 & \wedge & \texttt{f904008100008000}_{16} \end{pmatrix} \\ X &= \begin{pmatrix} L_1 & \wedge & \texttt{01040080}_{16} \\ L_8R_7 & \wedge & \texttt{2104008000008000}_{16} \end{pmatrix} \end{aligned}$$

with the notations used in Section C.2 and where L_1, L_8 and R_7 are computed from $P = (L_0, R_0)$ and $C = (L_8, R_8)$ using k [91]. It is easy to see that k' and S are sufficient to compute X. For reasons related to the structure of F, the masks on the subkeys are coded in octal while the masks on the message registers are coded in hexadecimal. We have $\ell = 2^{12}$ (this is the number of candidates K), $s = 2^{19}$ (number of possible samples S) and $q = 2^8$. The bias is approximated by tricky heuristic arguments to $|\delta| = 1.19 \times 2^{-21}$. Using $n = 2^{43}$, we have $\epsilon = 3.37$. Therefore, using two such characteristics as Matsui did (that is using it together with its reversed characteristic obtained by exchanging the left and the right masks), the global efficiency is given by the Equation (C.1) and the exhaustive search gets its complexity improved by a factor $\Phi(-3.37) = 2^{-11.4}$. The complexity of the Searching Phase is evaluated to $2^{44.6}$. (Matsui's experiment would have yielded complexity 2^{43}, so Theorem C.4 may be a little pessimistic, but we notice that the approximation $\epsilon \approx \sqrt{nq}d_2(D, D') = 3.78$ yields complexity $2^{42.38}$.)

Experiment	1	2	3	4	5	6	7	8	9	10
Matsui's attack	1	280	1	2	59	10	12	35	1	4
linear mark #1	1	46	1	2	205	48	8	58	2	4
linear mark #2	1	46	1	2	204	81	11	59	2	4
linear mark #3	1	100	1	2	205	48	8	60	2	6
linear mark #4	1	100	1	2	206	80	11	57	2	6
linear mark #5	1	45	1	2	103	79	8	58	2	4
linear mark #6	1	44	1	2	104	48	11	58	2	6
linear mark #7	1	101	1	2	104	48	11	57	2	6
linear mark #8	1	100	1	2	103	80	8	57	2	4
$\sum$(linear marks)^2	1	68	1	2	140	57	10	55	2	6
χ^2 attack	100	2221	516	197	435	1294	3667	2389	335	1320

Figure C.4 – Experiment of attacks on 8-rounds DES

C.5.3 Generalized Linear Test

We can generalize Matsui's statistic by any linear one using suitable a_x:

$$\Sigma_{\text{glin}} = \sum_{i=1}^{n} a_{X_i} = \sum_x a_x n_x.$$

Theorem C.7. *Σ_{glin} is asymptotically normal. The best efficiency is obtained with $a_x = v_k^x$. If D is uniform over $\mathcal{Q}$, it is*

$$\epsilon = \sqrt{nq}.d_2(D, D').$$

Proof. We have

$$\mu' - \mu = n \sum_x a_x v_k^x$$

and

$$\sigma = \sqrt{n}\sqrt{\sum_x (a_x - \bar{a})^2 \Pr_{X \in D}[X = x]}$$

where $\bar{a} = \sum_x a_x \Pr_{X \in D}[X = x]$. Σ_{glin} is asymptotically normal, due to the central limit Theorem [41]. Thus, we have

$$\epsilon^2 = n \frac{\left(\sum_x a_x v_k^x\right)^2}{\sum_x (a_x - \bar{a})^2 \Pr_{X \in D}[X = x]}.$$

The Theorem comes from Cauchy-Schwarz's Inequality in the particular case where $\Pr_{X \in D}[X = x] = \frac{1}{q}$. □

The problem of using the best linear statistic is similar to the problem of linear cryptanalysis, where we have to guess a vector a coming from a linear characteristic. Here, we have to bet on the transition matrix to get all a_x. If there are only few possible transition matrices, we use the sum of the squares

of all the linear marks as a new statistic, which turns out to be almost as efficient as the best one. (The reason why we use the sum-of-squares is that the different marks are linearly dependent, so a linear mean would be suject to strange cancellations.)

C.5.4 A slight Improvement of Matsui's Attack

For Matsui's symmetrized characteristic which is defined by

$$\begin{aligned} h_1(k) &= \begin{pmatrix} k_1 & \wedge & \texttt{0000000077000000}_8 \\ k_{16} & \wedge & \texttt{7700000000000000}_8 \end{pmatrix} \\ h_2(P,C) &= \begin{pmatrix} L_0R_0 & \wedge & \texttt{2104008000001f800}_{16} \\ L_8R_8 & \wedge & \texttt{f9040081000080000}_{16} \end{pmatrix} \\ h_3(k',S) &= \begin{pmatrix} L_1R_0 & \wedge & \texttt{2104008000008000}_{16} \\ L_8R_7 & \wedge & \texttt{2104008000008000}_{16} \end{pmatrix} \end{aligned}$$

we have $s = 2^{20}$, $\ell = 2^{12}$ and $q = 2^{10}$. Using the heuristic with projections, the deviation has been approximated to $d_2(D, D') \approx 2^{-24.58}$. Hence, using $2^{42.93}$ known plaintext/ciphertext couples (instead of $2^{43.00}$, which is 5% larger), we obtain $\epsilon = 3.69$. With two such characteristics, the exhaustive search is improved by a factor $\Phi(-3.69) = 2^{-13.14}$ and the Searching Phase gets a complexity $2^{42.86}$. Since off-line exhaustive search is cheaper than getting a new sample, we can afford 2^{42} known plaintexts which gives $\epsilon = 3.78$ then $2^{56}.\Phi(-3.78) = 2^{47.93}$: 2^{42} known plaintexts enables to find the key within a 2^{48} average complexity.

For eight-rounds DES, Matsui announced 1.49×2^{17} knwon plaintexts, but it was to get the same complexity than for sixteen-rounds DES in the exhaustive search, that is 2^{43}. Here, we have $d_2(D, D') \approx 2^{-11.86}$, so, with 2^{17} known plaintexts, we have $\epsilon = 3.11$ and the exhaustive search has complexity $2^{56}.\Phi(-3.11) = 2^{45.93}$ with two characteristics. (With 2^{18} known plaintexts, the same computation yields complexity $2^{38.50}$.) This attack has been implemented.

Experiments show there are only eight kinds of bias vector V_K. We use as a statistic the sum-of-the-squares of the eight marks obtained with the eight corresponding linear statistics. With the only characteristic defined in this Section, we have $\ell = 2^{12}$ candidates and the rank of the good candidate in the sorted list should be $1+\ell.\Phi(-\epsilon/\sqrt{2})$ on average. For $n = 2^{17}$ samples, we have $\epsilon = 3.11$ so the average rank should be 57.86. Ten random experiments yielded ranks illustrated on Figure C.4. We put ranks obtained by Matsui's mark, by each of the eight linear marks, by the sum-of-squares of the linear marks, and by the χ^2 mark we will present on next Section. This shows

the use of the best linear statistic slightly improves Matsui's attack. It also confirms that the χ^2 attack is a little less efficient than the other attacks, as we will prove.

C.6 χ^2 Cryptanalysis

The deviation from the uniform distribution in a domain with cardinality q can be tested using the χ^2 test [41]:

$$\Sigma_{\chi^2} = \frac{q}{n}\sum_x \left(n_x - \frac{n}{q}\right)^2 = \frac{q}{n}\sum_x {n_x}^2 - n.$$

Theorem C.8. *Under the hypothesis, the efficiency of the attack using Σ_{χ^2} is*

$$\epsilon \approx n\frac{q}{\sqrt{2(q-1)}}(d_2(D,D'))^2.$$

Proof. For bad candidates, the statistic Σ_{χ^2} tends to the χ^2 distribution with $q-1$ degrees of freedom: $\mu = q-1$ and $\sigma = \sqrt{2(q-1)}$. When the degree of freedom is large, this distribution can be approximated by a normal one.

Let

$$\Sigma'_{\chi^2} = \frac{q}{n}\sum_x \left(n_x - \frac{n}{q} - nv_k^x\right)^2.$$

Σ'_{χ^2} is a kind of χ^2 statistic such that

$$E(\Sigma'_{\chi^2}) = q - 1 - q(d_2(D,D'))^2.$$

So, we have

$$\Sigma_{\chi^2} = \Sigma'_{\chi^2} + 2d_2(D,D').\sqrt{qn}\Sigma_{\text{glin}} - nq(d_2(D,D'))^2$$

where Σ_{glin} is the best standardized linear test (*i.e.* with $E(\Sigma_{\text{glin}}) = 0$ and $\sigma(\Sigma_{\text{glin}}) = 1$). So, we have

$$\mu' = q - 1 + (n-1)q(d_2(D,D'))^2$$

which allows to compute ϵ. □

A straightforward consequence of this Theorem is that with the same characteristic and the same number a plaintext/ciphertext couples, the χ^2 cryptanalysis is more efficient than the differential cryptanalysis (which is a quadratic statistic).

Using this statistic, we do not need to have a precise idea of which information is leaked throughout Enc, such as what would have been done in linear or differential cryptanalysis using a particular vector a. Here, we use a characteristic, and if there exists a powerful subcharacteristic according to linear of differential cryptanalysis, the χ^2 test is able to detect it and to use it to distinguish the good k'.

For instance, we can try Matsui's symmetrized characteristic with the χ^2 cryptanalysis. We have $q = 2^{10}$ and $d_2(D, D') \approx 2^{-24.58}$. Using $2^{46.2}$ known plaintexts (9 times as Matsui does), we get $\epsilon = 2.90$. Here the χ^2 variable is approximately normal. So, using two such characteristics, we get the average complexity $2^{56}.\Phi(-2.90) = 2^{46.9}$.

C.7 Conclusion

We have shown that differential and linear cryptanalysis can be viewed in a more statistical approach. It is possible to join the efforts of several characteristics to improve them. Both attacks can be improved using an additional information, that is the vector of all v_k^x. Conversely, with less knowledge about the characteristic (that is without the precise knowledge of which bits of the input and the output play a role and what happens in between), the χ^2 cryptanalysis performs an attack which is roughly as efficient.

To prove that the linear aspects of differential or linear cryptanalysis are not unavoidable, we presented a new heuristic method which has produced the same attack than Matsui's. This leads to new directions in cryptanalysis. We hope that this new approach and the experiments presented in this paper will motivate further investigations in the use of statistic experiments in cryptanalysis.

Acknowledgment

We wish to thank Eli Biham, Don Coppersmith, Carlo Harpes, Lars Knudsen and Jacques Stern for fruitful discussions.

Annexe D

On the Need for Multipermutations: Cryptanalysis of MD4 and SAFER

[*Cet article de* Serge Vaudenay *a été publié dans les actes du colloque* Fast Software Encryption *94 [162].*]

Abstract. Cryptographic primitives are usually based on a network with boxes. At EUROCRYPT'94, Schnorr and the author of this paper claimed that all boxes should be multipermutations. Here, we investigate a few combinatorial properties of multipermutations. We argue that boxes which fail to be multipermutations can open the way to unsuspected attacks. We illustrate this statement with two examples.

Firstly, we show how to construct collisions to MD4 restricted to its first two rounds. This allows one to forge digests close to each other using the full compression function of MD4. Secondly, we show that variants of SAFER are subject to attack faster than exhaustive search in 6.1% cases. This attack can be implemented if we decrease the number of rounds from 6 to 4.

In [146], *multipermutations* are introduced as formalization of perfect diffusion. The aim of this paper is to show that the concept of multipermutation is a basic tool in the design of dedicated cryptographic functions, as functions that do not realize perfect diffusion may be subject to some clever cryptanalysis in which the flow of information is controlled throughout the computation network. We give two cases of such an analysis.

Firstly, we show how to build collisions for MD4 restricted to its first two rounds[1]. MD4 is a three rounds hash function proposed by Rivest[128]. Den Boer and Bosselaers[32] have described an attack on MD4 restricted to its last two rounds. Another unpublished attack on the first two rounds has been found by Merkle (see the introduction of [32]). Here, we present a new attack which is based on the fact that an inert function is not a multipermutation. This attack requires less than one tenth of a second on a SUN workstation. Moreover, the same attack applied to the full MD4 compression function produces two different digests close to each other (according to the Hamming distance). This proves the compression function is not correlation-free in the sense of Anderson[15].

Secondly, we show how to develop a known plaintext attack to a variant of SAFER K-64, in which we replace the permutation $\exp_{45}$ by a (weaker) one. SAFER is a six rounds encryption function introduced by Massey[87]. It uses a byte-permutation (namely, $\exp_{45}$ in the group of nonzero integers modulo 257) for confusion. If we replace $\exp_{45}$ by a random permutation P (and $\log_{45}$ by P^{-1}), we show that in 6.1% of the cases, there exists a known plaintext attack faster than exhaustive search. Furthermore, this attack can be implemented for the function restricted to 4 rounds. This attack is based on the linear cryptanalysis introduced by Matsui[89] and recently gave way to the first experimental attack of the full DES function[91].

D.1 Multipermutations

In [146], multipermutations with 2 inputs and 2 outputs are introduced. Here, we propose to generalize to any number of inputs and outputs.

Definition D.1. *A (r, n)-multipermutation over an alphabet Z is a function f from Z^r to Z^n such that two different $(r+n)$-tuples of the form $(x, f(x))$ cannot collide in any r positions.*

Thus, a $(1, n)$-multipermutation is nothing but a vector of n permutations over Z. A $(2, 1)$-multipermutation is equivalent to a *latin square*[2]. A $(2, n)$-multipermutation is equivalent to a set of n two-wise *orthogonal* latin squares[3]. Latin squares are widely studied by Dénes and Keedwell in [45].

An equivalent definition says that the set of all $(r+n)$-tuples of the form $(x, f(x))$ is an error correcting code with minimal distance $n+1$, which is

[1]This part of research has been supported by the CELAR.

[2]a latin square over a finite set of k elements is a $k \times k$ matrix with entries from this set such that all elements are represented in each column and each row.

[3]two latin squares A and B are orthogonal if the mapping $(i, j) \mapsto (A_{i,j}, B_{i,j})$ gets all possible couples.

the maximal possible. In the case of a linear function f, this is the definition of MDS codes: codes which reach Singleton's bound (for more details about MDS codes, see [94]). More generally, a (r, n)-multipermutation is equivalent to a $((\#Z)^r, r+n, \#Z, r)$-orthogonal array[4].

A multipermutation performs a *perfect diffusion* in the sense that changing t of the inputs changes at least $n - t + 1$ of the outputs. In fact, it corresponds to the notion of *perfect local randomizer* introduced by Maurer and Massey[97] with the optimal parameter. If a function is not a multipermutation, one can find several values such that both few inputs and few outputs are changed. Those values can be used in cryptanalysis as is shown in two examples below. This motivates the use of multipermutations in cryptographic functions.

The design of multipermutations over a large alphabet is a very difficult problem, as the design of two-wise orthogonal latin squares is a well-known difficult one. The only powerful method seems to use an MDS code combined with several permutations at each coordinate.

In the particular case of 2 inputs, it is attractive to choose latin squares based on a group law: if we have a group structure over Z, we can seek permutations α, β, γ, δ, ϵ and ζ such that

$$(x, y) \mapsto (\alpha[\beta(x).\gamma(y)], \delta[\epsilon(x).\zeta(y)])$$

is a permutation, as it will be sufficient to get a multipermutation. Unfortunately, it is possible to prove that such permutations exist only when the 2-Sylow subgroup of Z is not cyclic[5], using a theorem from Hall and Paige[58]. More precisely, they do not exist when the 2-Sylow subgroup is cyclic. They are known to exist in all solvable groups in which the 2-Sylow subgroup is not cyclic, but the existence in the general case is still a conjecture. Hence, Z should not have a cyclic group structure. For instance, we can use the $\mathbf{Z}_2^n$ group structure for $n > 1$. Such multipermutations are proposed in [146].

In MD4, the group structure of $\mathbf{Z}_2^{32}$ is used, but some functions are not multipermutations. On the other hand, in SAFER, the group structure of $\mathbf{Z}_{256}$, which is cyclic, is used, so without multipermutations.

[4] a $(M, r+n, q, r)$-orthogonal array is a $M \times (r+n)$ matrix with entries from a set of q elements such that any set of r columns contains all q^r possible rows exactly $\frac{M}{q^r}$ times.

[5] we agree the trivial group is not cyclic. Actually, $x \mapsto x^2$ is an orthomorphism in all groups with odd order, in which the 2-Sylow subgroup is trivial.

D.2 Cryptanalysis of MD4

D.2.1 Description of MD4

MD4 is a hash function dedicated to 32-bit microprocessors. It hashes any bit string into a 128-bit digest. The input is padded following the Merkle-Damgård scheme[43, 102] and cut into 512-bit blocks. Then, each block is processed iteratively using the Davies-Meyer scheme[44, 96] i.e. with an encryption function C in a feedforward mode: if $B_1, \ldots, B_n$ is the sequence of blocks (the padded message), the hash value is

$$h_{B_n}(\ldots h_{B_1}(v_i)\ldots)$$

where v_i is an Initial Value, and $h_x(v)$ is $C_x(v) + v$ (x is the key and v is the message to encrypt).

Here we intend to build a single block collision to $h(v_i)$, that is to say two blocks x and x' such that $C_x(v_i) = C_{x'}(v_i)$. It is obvious that this can be used to build collisions to the hash function. So, we only have to recall the definition of the function $C_x(v)$.

The value v is represented as 4 integers a, b, c and d (coded with 32 bits), and the key x is represented as 16 integers $x_1, \ldots, x_{16}$. The initial definition of C uses three rounds $i = 1, 2, 3$. The figure D.1 shows the computational graph of a single round i. It uses a permutation σ_i and some boxes B_i^j. B_i^j is fed with a main input, a block integer $x_{\sigma_i(j)}$ and three side inputs. If p is the main input and q, r and s are the side inputs (from top to bottom), the output is

$$R^{\alpha_{i,j}}(p + f_i(q, r, s) + x_{\sigma_i(j)} + k_i)$$

where R is the right circular rotation, $\alpha_{i,j}$ and k_i are constants and f_i is a particular function. In the following, we just have to know that f_2 is the bit-wise majority function, σ_1 is the identity permutation, and

$$\sigma_2 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\ 1 & 5 & 9 & 13 & 2 & 6 & 10 & 14 & 3 & 7 & 11 & 15 & 4 & 8 & 12 & 16 \end{pmatrix}$$

D.2.2 Attack on the First Two Rounds

If we ignore the third round of C, it is very easy to build collisions. We notice that no B_2^j are multipermutations: if $p = 0$, $x = -k_2$ and two of the three integers q, r and s are set to zero, then $B_2^j(x, p, q, r, s)$ remains zero (the same remark holds with -1 instead of 0). So, we can imagine an attack where two

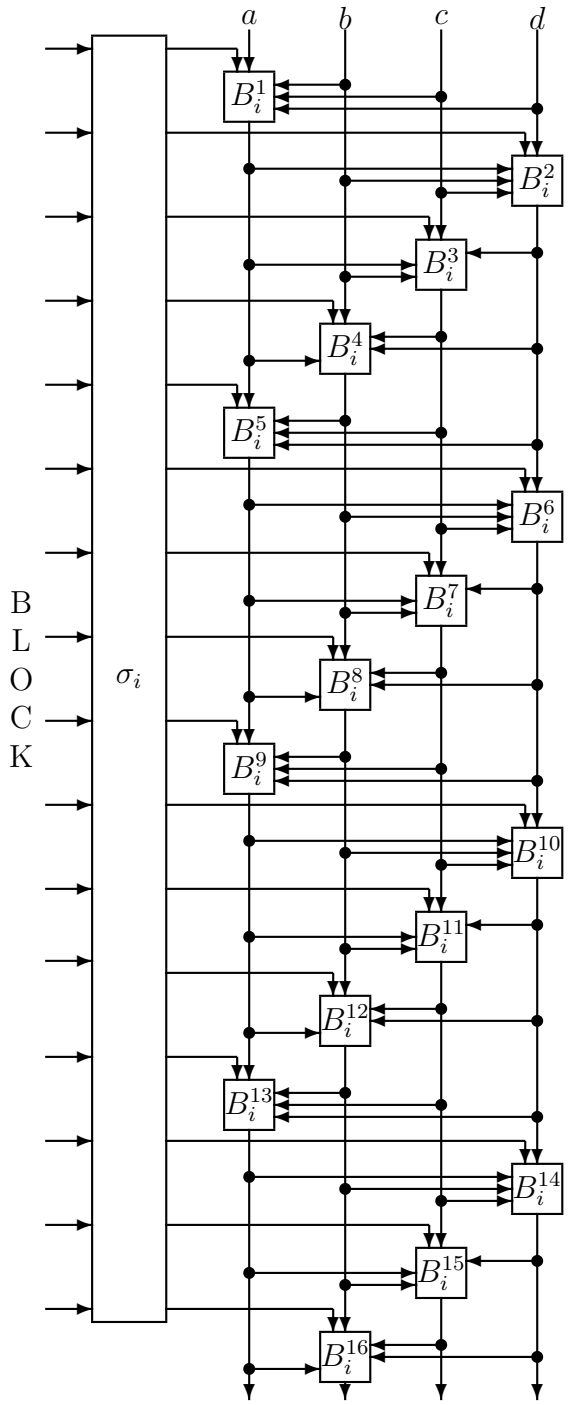


Figure D.1 – One round of C

blocks differ only in x_{16}, the other integers are almost all set to $-k_2$ and such that almost all the outputs of the first round are zero. This performs a kind of corridor where the modified values are controlled until the final collision.

More precisely, let $x_1, \ldots, x_{11}$ equal $-k_2$, x_{12} be an arbitrary integer (your phone number for instance) and x_{13}, x_{14} and x_{15} be such that the outputs a, c and d of the first round are zero. The computation of x_{13}, x_{14} and x_{15} is very easy from the computational graph. Thanks to the previous remark, we can show that the outputs a, c and d of the second round do not depend on x_{16} as the modified information in x_{16} is constrained in the register b. Thus, modifying x_{16} does not modify a, c and d.

Letting the b output be a function of x_{16}, we just have to find a collision to a 32 bits to 32 bits function. This can be done very efficiently using the birthday paradox or the ρ method. An implementation on a Sparc Station uses one tenth of second.

If we use the same attack on the full-MD4 function, since $\sigma_3^{-1}(16) = 16$, the only modified x occurs in the very last computation in the third round. So, if this round is fed with a collision, it produces a collision on the a, c and d output. The digests differ only in the second integer b. Hence, the average Hamming distance between both digests is 16. This proves the compression function of MD4 is not correlation-free, according to Andersons's definition[15].

D.3 Cryptanalysis of SAFER

D.3.1 Description of SAFER

SAFER is an encryption function dedicated to 8-bit microprocessors. It encrypts a 64 bits message using a 64-bit key. The key is represented as 8 integers $k_1, \ldots, k_8$. A key scheduling algorithm produces several subkeys $k_1^i, \ldots, k_8^i$. In the following, we just have to know that k_j^i is a simple function of k_j (and $k_j^1 = k_j$).

The encryption algorithm takes 6 rounds and a half. The ith round is summarized on figure D.2. It uses the subkeys k_j^{2i-1} and k_j^{2i}. After the 6th round, the half round simply consists of xoring/adding the subkeys k_j^{13} as we would do in a 7th round.

$\oplus$ represents the xor operation on 8 bits integers. $+$ is the addition modulo 256. P is a permutation over the set of all 8-bits integers defined in the SAFER design. Q is the inverse permutation of P. L is a linear permutation

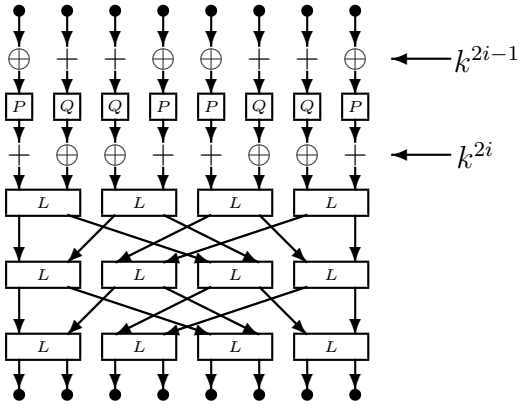


Figure D.2 – The ith round of SAFER

over the algebraic structure of the ring $\mathbf{Z}_{256}$, as

$$L(x, y) = (2x + y, x + y) \pmod{256}.$$

In the original design, P is the exponentiation in base 45 modulo 257: all integers from 1 to 256 can be coded with 8 bits (256 is coded as zero) and represent the group of all invertible integers modulo 257. 45 is a generator of this group.

In practical implementations, we have to store the table of the permutation P. So, there is no reason to study SAFER with this particular permutation. Here, we will show that this choice is a very good one, as for 6.1% of all possible permutations, there exists a known plaintext attack faster than exhaustive search.

D.3.2 Linear Cryptanalysis of SAFER

> *I have many times used the discrete exponential or the discrete logarithm as nonlinear cryptographic functions and they have never let me down.*
> James Massey

The permutation L is not a multipermutation, as we have

$$L_1(x + 128, y) = L_1(x, y)$$

for all x and y (where L_1 denotes the first output of L). So, we have pairs of 4-tuples $(x, y, L(x, y))$ at Hamming distance 2. Actually, there are no $(2, 2)$-multipermutations which are linear over $\mathbf{Z}_{256}$ as its 2-Sylow subgroup is cyclic

(it is itself here). We can use this property of L_1 by a dual point of view noticing that some information about $L_1(x, y)$ only depends on y. Namely, we have

$$L_1(x, y) \cdot 1 = y \cdot 1$$

where $\cdot$ is the inner product over $(\mathbf{Z}_2)^8$, so, $y \cdot 1$ is the least significant bit of y. Similarly, we have

$$(L_1(x, y) \cdot 1) \oplus (L_2(x, y) \cdot 1) = x \cdot 1$$

Let F denote the function defined by the three bottom layers on figure D.2 (layers which uses L in a round). If $x_1, \ldots, x_8$ are the inputs of a round, the outputs are $F(y_1, \ldots, y_8)$ where $y_1 = P(x_1 \oplus k_1^1) + k_1^2$, ... We notice that if $F(y_1, \ldots, y_8) = (z_1, \ldots, z_8)$, we have a 2-2 *linear characteristic*

$$(z_3 \cdot 1) \oplus (z_4 \cdot 1) = (y_3 \cdot 1) \oplus (y_4 \cdot 1)$$

(this means there is a linear dependence using 2 inputs and 2 outputs of F). There are 5 other 2-2 linear characteristics:

$$\begin{aligned}
(z_2 \cdot 1) \oplus (z_6 \cdot 1) &= (y_2 \cdot 1) \oplus (y_6 \cdot 1) \\
(z_5 \cdot 1) \oplus (z_7 \cdot 1) &= (y_5 \cdot 1) \oplus (y_7 \cdot 1) \\
(z_3 \cdot 1) \oplus (z_7 \cdot 1) &= (y_5 \cdot 1) \oplus (y_6 \cdot 1) \\
(z_5 \cdot 1) \oplus (z_6 \cdot 1) &= (y_2 \cdot 1) \oplus (y_4 \cdot 1) \\
(z_2 \cdot 1) \oplus (z_4 \cdot 1) &= (y_3 \cdot 1) \oplus (y_7 \cdot 1).
\end{aligned}$$

If L were a multipermutation, the smallest characteristics would be a-b ones such that $a + b = 6$. This property is similar to the well-known Heisenberg's inequality which states we cannot have any precise information on both the input and the output of the Fourier transform. This means more information would be required in a cryptanalysis.

Let q denote $\text{Prob}_x[x \cdot 1 = P(x) \cdot 1] - \frac{1}{2}$, the bias which measures the dependence between the least lignificant bits of $P(x)$ and x. We get the same bias with Q in place of P. If $(x_1, \ldots, x_8)$ is a plaintext, if $y_1 = P(x_1 \oplus k_1)$, $y_2 = Q(x_2 + k_2)$, ..., $y_8 = P(x_8 \oplus k_8)$, and if $(z_1, \ldots, z_8)$ is the ciphertext, let us write

$$b(x, z) = (y_3 \cdot 1) \oplus (y_4 \cdot 1) \oplus (z_3 \cdot 1) \oplus (z_4 \cdot 1).$$

Lemma D.3 in appendix D.A states that $b(x, z) = \phi(k)$ with probability $\frac{1}{2}(1 + (2q)^{10})$ where $\phi(k)$ denotes the exclusive or of all least significant bits of k_3^i and k_4^i for $i = 2, \ldots, 13$. For a given (x, z), to compute $b(x, z)$, we only

have to know k_3 and k_4. Lemma D.3 states that it occurs with probability roughly equal to $\frac{1}{2}$ (the difference with $\frac{1}{2}$ is negligible against $(2q)^{10}$) when wrong k_3 and k_4 are used in the computation of $b(x,z)$. Thus, trying all the possible (k_3,k_4), it is possible to distinguish the good one from the other candidates by a statistical measure.

Let us recall the central limit theorem (see [52] for instance):

Theorem D.2. *If B is the arithmetic mean of N independent random variables with the same probability distribution with average μ and standard deviation σ, we have*

$$Prob\left[(B-\mu)\frac{\sqrt{N}}{\sigma}\in[a,b]\right]\to\frac{1}{\sqrt{2\pi}}\int_a^b e^{-\frac{t^2}{2}}dt.$$

Let $B(k_3,k_4)$ be the average of $b(x,z)$ over all the N available couples (x,z). Lemma D.3 proves that the standard deviation of $b(x,z)$ is close to $\frac{1}{2}$. Let

$$\lambda_1+\lambda_2=\sqrt{N}(2q)^{10}.$$

The central limit theorem states that if (k_3,k_4) is wrong,

$$\text{Prob}\left[\left|B(k_3,k_4)-\frac{1}{2}\right|<\frac{\lambda_1}{2\sqrt{N}}\right]\to\frac{1}{\sqrt{2\pi}}\int_{-\lambda_1}^{\lambda_1}e^{-\frac{t^2}{2}}dt;$$

and if (k_3,k_4) is good,

$$\text{Prob}\left[\left|B(k_3,k_4)-\frac{1}{2}\right|<\frac{\lambda_1}{2\sqrt{N}}\right]\to\frac{1}{\sqrt{2\pi}}\int_{\lambda_2}^{\lambda_2+2\lambda_1}e^{-\frac{t^2}{2}}dt.$$

The statistical test consists in accepting any (k_3,k_4) such that

$$\text{Test}(k_3,k_4):\qquad\left|B(k_3,k_4)-\frac{1}{2}\right|>\frac{\lambda_1}{2\sqrt{N}}.$$

If $\lambda_1=\lambda_2=2$ the good (k_3,k_4) is accepted with probability 98% and the bad ones are rejected with probability 95%. So, the number of plaintexts/ciphertexts required to distinguish the good (k_3,k_4) is

$$N\sim\frac{16}{(2q)^{20}}.$$

If $|q|$ is greater than 2^{-4}, this is faster than exhaustive search.

For only 4 rounds in SAFER, we have $N\sim\frac{16}{(2q)^{12}}$. So, for all permutations P which are biased ($q\neq 0$), this attack is faster than exhaustive search. For $|q|\geq 2^{-4}$, the attack can be implemented.

The analysis of the distribution of q shows that we have $|q| \geq 2^{-4}$ for 6.1% of the possible permutations P (see appendix D.B). We have $q = 0$ for only 9.9% of the permutations. Unfortunately (or fortunately), for the P chosen by Massey, we have $q = 0$, so, the weakness of the diffusion phase is balanced by the strength of the confusion phase. Actually, $q = 0$ is a property of all exponentiations which are permutations (see appendix D.C). This analysis illustrates how Massey was right in context with his quotation.

Further analysis can improve this attack. It is possible to use tighter computations. We can look for a better tradeoff between the workload and the probability of success. It is also possible to use several characteristics to decrease N (for more details, see [71]). At least, it is possible to decrease N by a factor of 64. Actually, we believe it is possible to improve successfully this attack for all the 90.1% biased permutations.

Conclusion

In MD4, we have shown that the fact that f_2 is not a multipermutation allows one to mount an attack. Similarly, in SAFER, the diffusion function is not a multipermutation. This allows one to imagine another attack. This shows that we do need multipermutations in the design of cryptographic primitives. Research in this area should be motivated by this general statement.

Acknowledgments

I would like to thank Antoon Bosselaers, Ross Anderson, James Massey and Carlo Harpes for helpful information. I thank the CELAR for having motivated the research on MD4. I also thank Jacques Stern and Hervé Brönnimann for their help.

D.A Linear Characteristic

Lemma D.3. *If $\phi(k)$ denotes the least lignificant bit of the sum of all k_3^i and k_4^i for $i = 2, \ldots, 13$, let us denote $y_3 = Q(x_3 + k_3)$, $y_4 = P(x_4 \oplus k_4)$ and*

$$b(x, z) = (y_3 \cdot 1) \oplus (y_4 \cdot 1) \oplus (z_3 \cdot 1) \oplus (z_4 \cdot 1)$$

where z is the encrypted message of x using an unknown key. $b(x, z) = \phi(k)$ holds with probability

$$\frac{1}{2}(1 \pm (2q)^{10+e})$$

where e is the number of wrong integers in (k_3, k_4) ($e = 0$ if both are good and $e = 2$ in most of cases). The standard deviation of $b(x, z)$ is

$$\frac{1}{2}\sqrt{1 - (2q)^{20+2e}}.$$

Proof. Thanks to the property of the linear characteristic, if we denote by t_j^i the xor of the least significant bit of the input and the output of the P/Q box in position j in round $\#i$, it is easy to se that

$$b(x, z) = (y_3 \cdot 1) + (y_4 \cdot 1) + (y_3' \cdot 1) + (y_4' \cdot 1) + \sum_{i=2}^{6}\sum_{i=3}^{4} t_j^i + \phi(k') \pmod 2$$

where $\phi(k')$ denotes the real $\phi(k)$ and y_3' (resp. y_4') denotes the real y_3 (resp. y_4). Under the heuristic assumption that all inputs to P/Q boxes are uniformly distributed and independent, it is easy to prove that

$$\text{Prob}\left[\sum_{i=2}^{6}\sum_{i=3}^{4} t_j^i = 0\right] = \frac{1}{2}(1 + (2q)^{10})$$

using the piling-up lemma pointed out by Matsui[89]. This finishes the case where k_3 and k_4 are good.

If k_3 or k_4 are wrong, let us denote $e = 2$ if both are bad, and $e = 1$ if only one is bad. Assume k_3 is bad without loss of generality. We have

$$\text{Prob}\left[(y_3 \cdot 1) \oplus (y_3' \cdot 1) = 0\right] = \frac{1}{2}(1 + 2q).$$

The $\pm$ comes from whether $\phi(k) = \phi(k')$ or not. This finishes the computation of the probability.

The standard deviation comes from the following formula which holds for all 0/1 random variables :

$$\sigma(b) = \sqrt{E(b)(1 - E(b))}.$$

□

D.B Distribution of the Bias

Lemma D.4. *If $q = Prob[x \cdot 1 = P(x) \cdot 1] - \frac{1}{2}$ where P is a permutation over $\{0, \ldots, n-1\}$ (we assume that n is a multiple of 4), nq is always an even integer and for all integers k*

$$Prob\left[q = \frac{2k}{n}\right] = \frac{\left((\frac{n}{2})!\right)^4}{n!\left(\left(\frac{n}{4} - k\right)!\right)^2\left(\left(\frac{n}{4} + k\right)!\right)^2}$$

for a uniformly distributed permutation Pd.

All those kind of distribution has been studied by O'Connor, but we give here and independant study in this particular case [116].

Proof. If $k+\frac{n}{4}$ denotes the number of even integers x such that $P(x)$ is even, we have $q = \frac{2k}{n}$. So, we just have to enumerate the number of permutations for a given $k + \frac{n}{4}$.

We have to choose 4 sets with $k + \frac{n}{4}$ elements in sets with $\frac{n}{2}$ elements: the set of even integers which are mapped on even integers, the set of their images, the set of odd integers which are mapped on odd integers and the set of their images. We also have to choose 2 permutations over a set of $k + \frac{n}{4}$ integers (how to connect even to even integers and odd to odd integers) and 2 permutations over a set of $-k + \frac{n}{4}$ integers (how to connect even to odd integers and odd to even integers). So, the number of permutations is

$$\left(\begin{array}{c} \frac{n}{2} \\ \frac{n}{4}+k \end{array} \right)^4 \times \left(\left(\frac{n}{4}+k \right)! \right)^2 \times \left(\left(\frac{n}{4}-k \right)! \right)^2 .$$

□

This allows one to compute

$$\text{Prob}\left[|q| \geq 2^{-4} \right] \simeq 6.1\%$$

for $n = 256$ and

$$\text{Prob}\left[q = 0 \right] \simeq 9.9\%.$$

D.C Bias of the Exponentiation

Lemma D.5. *For any generator g of $\mathbf{Z}_{257}^*$, the permutation $x \mapsto g^x$ is unbiased (i.e. $q = 0$).*

Proof. We have $(g^{128})^2 \equiv g^{256} \equiv 1 \pmod{257}$ so g^{128} is 1 or -1. As the exponentiation in base g is a permutation and $g^0 = 1$, we have $g^{128} \equiv -1 \pmod{257}$.

We have $g^{x+128} \equiv -g^x \equiv 257 - g^x \pmod{257}$, so, we can partition all the integers into pairs $\{x, x+128\}$ of integers with the same least significant bit. The image of this pair by the exponentiation has two different least significant bits, so the bias q is 0. □

Annexe E

The Black-Box Model for Cryptographic Primitives

[*Cet article de* Claus Schnorr *et* Serge Vaudenay *a été publié dans le* Journal of Cryptology *en 1998 [147]. Il effectue la synthèse des articles parus dans* Fast Software Encryption 93 *[145] et Eurocrypt 94 [146].*]

[**Erratum.** Theorem E.10 p. 156 should read

$$C(G_{k,s}^{\frac{1}{2}d}) \leq 2^{k-2} + 2^{s-k}.$$

($s - k$ has been swapped at the end of the proof.)]

Abstract. We introduce the *black-box model* for cryptographic primitives. In this model cryptographic primitives are given by a computation graph, where the computation boxes sitting on the vertices of the graph act as random oracles. We formalize and study a family of generic attacks which generalize exhaustive search and the birthday paradox. We establish complexity lower bounds for these attacks and we apply it to compression functions based on the FFT network.

Introduction. Cryptographic primitives for encryption, hashing and pseudorandom generation are judged according to efficiency and security. Design methods for constructing cryptographic primitives are usually empiristic. The example of Rivest's MD4 hash function [128], which has been shown insecure by Dobbertin [48], demonstrates the need for a design theory that provides security in a realistic model.

Usually cryptographic primitives are defined as a computation graph in which the vertices are computation boxes. The cryptanalysis approach of Biham-Shamir [30] and Matsui [91] initiated an important study of the algebraic properties of the computation boxes. In this paper we take another view, we neglect the inner structure of the boxes. We study the security provided by a computation graph, where the boxes act as random oracles, i.e. as *black-boxes*.

Generic attacks that do not exploit the inner structure of the boxes play an important role. Nechaev [109] and Shoup [151] study generic algorithms for the discrete logarithm, assuming that the group operations are given as black-boxes. The random oracle model of Bellare and Rogaway [21] has been used in security proofs for various signature schemes. Here the hash function is a black-box acting as a random oracle in a network comprising the signer, the verifier and the attacker of a signature scheme. We propose a black-box model in which *all* boxes of the computation graph act as random oracles.

The black-box model covers powerful attacks, e.g. the iterative use of exhaustive search and the birthday paradox applied locally to arbitrary parts of the computation graph. We only exclude attacks that "split" the boxes. In the black-box model we can prove interesting and tight complexity bounds for generic attacks. These complexities correspond to the minimal workload of attacks. We study the average complexity of these attacks for relevant probability distributions for the boxes. Black-box cryptanalysis can determine optimal interconnection networks for the design of hash functions and symmetric encryption functions provided that strong computation boxes are given.

In particular we extend and analyse the FFT network, the computation graph of the Fast Fourier Transform that is known as the butterfly graph. This network has been used in several cryptographic proposals [87, 145, 146]. We give evidence that the FFT network with 2^k input nodes and $2k-2$ layers yields a family of compression functions with optimal security in the black-box model. This means that for the doubled FFT network there is no better black-box attack then exhaustive search over the inputs. While our lower and upper complexity bounds coincide for *linear* attacks, lower and upper bounds differ by a factor 3 in the general case.

In Section E.1 we present the black-box model. In Section E.2 we consider the FFT hash network. We prove upper bounds and lower bounds for the complexity of inverting the function computed by this network. The particular case of linear algorithms for the FFT network has been studied in [146]. Here we prove lower bounds for general algorithms in the black-box model. A full formal study is also available in [163, pp. 31–87].

E.1 The Black-Box Model

E.1.1 Computation Graphs with Random Boxes

Let $G = (V, E)$ be an undirected graph with vertex set V and edge set E.[1] A "computation" along G associates with each edge a value in some finite *alphabet* Z. Associated with each vertex v is a set of possible local computations (or local solutions) $I(v) \subset Z^{E(v)}$ for $E(v)$, the set of edges adjacent to v. Thus $I(v)$ is the set of assignments of values in Z to the edges in $E(v)$ that are admissible for the box at v. For the degree $d(v) = \#E(v)$[2] of vertex v we must have $\#I(v) \leq \#Z^{d(v)}$. If $\#I(v) = \#Z^{d(v)}$ the box is trivial as all assignments are admissible. If vertex v has i "input edges" then $\#I(v) = \#Z^i$ since the input values determine the output values. A *solution* for the graph is a tuple $t \in Z^E$, i.e. a tuple on E such that for all vertices v, the restriction $t_{|E(v)}$[3] is in $I(v)$.

We call I — the collection of all local solutions — an *interpretation* of G. We study resolution algorithms that work in general, i.e. for random local solutions. We study the average complexity of these algorithms. We associate with each vertex v a *degree of freedom* $\mathrm{df}(v)$ — informally the (fractional) number of independent values in Z that appear in $I(v)$ — which we define as $\log_{\#Z} \mathrm{Exp}_I \#I(v)$[4]. In the following all logarithms have the basis $\#Z$ and log means $\log_{\#Z}$.

Definition E.1. *A* computation graph (G^{df}, Z) *consists of an undirected graph* $G = (V, E)$*, a real valued function* $\mathrm{df}(v)$ *satisfying* $\mathrm{df}(v) \leq d(v)$ *and an alphabet* Z*. A* random interpretation I *is a random map which associates with each vertex* v *a set* $I(v) \subset Z^{E(v)}$ *of local solutions so that* $\mathrm{df}(v) = \log \mathrm{Exp}_I \#I(v)$.

The computation graph G^{df} is undirected and so is the "computation flow". To stress the undirected nature of G^{df} it was called *equation graph* rather than *computation graph* in [163].

In the following we assume that all probability distributions for I have the following two properties:

Local uniformity. *For all* $v \in V$ *and* $t \in Z^{E(v)}$*, the probability* $\Pr[t \in I(v)]$ *is a constant which depends on* v*. Thus we have* $\Pr[t \in I(v)] = \#Z^{\mathrm{df}(v)-d(v)}$.

[1] All graphs are finite and loop-free in the paper.

[2] The symbol # denotes the cardinality of a set.

[3] The notation $t_{|E'}$ is the restriction of the tuple t to the subset E', i.e. the tuple on E' which is equal to t on all entries in E'.

[4] Here Exp_I denotes the expected value over the distribution of I.

Independence. *The sets $I(v)$ are independent for distinct vertices v.*

Examples of possible distributions are

- the uniform distribution over all I so that $I(v)$ is a subset of $Z^{E(v)}$ with $\#I(v) = \#Z^{\mathrm{df}(v)}$;
- for integer degree of freedom $\mathrm{df}(v)$, the uniform distribution over all I so that $I(v)$ defines a function of edge values of some $\mathrm{df}(v)$ edges in $E(v)$ to the other $d(v) - \mathrm{df}(v)$ edge values ;
- the uniform distribution over all I so that $I(v)$ defines a *multipermutation* on $Z^{E(v)}$. (Following Vaudenay [162, 163], a multipermutation with r inputs and n outputs is a set of $(r+n)$-tuples such that no different tuples t_1 and t_2 can be simultaneously equal on any r different entries. This way, it is a function of any r entries onto the remaining n ones. This concept formalizes the notion of perfect diffusion, that all inputs are perfectly diffused to the outputs in an information theoretic sense. This is a useful design criterion.)

E.1.2 Resolution and Complexity of a Computation Graph

For two subsets $E', E'' \subset E$ of edges and corresponding sets of tuples $X' \subset Z^{E'}$, $X'' \subset Z^{E''}$ we define the *join*

$$X' \bowtie X'' \;=\; \{t \in Z^{E' \cup E''} \mid t_{|E'} \in X' \,,\, t_{|E''} \in X''\}$$

to be the set of all tuples t on $E' \cup E''$ with restrictions to E' in X' and to E'' in X''. This operation $\bowtie$ is similar to the join used in the relational database theory. The join is associative and commutative and the set of *global* solutions on E turns out to be the join of all $I(v)$.

We extend the interpretation I to arbitrary subsets of vertices using that the join is associative and commutative. For $U \subset V$ we let $E(U)$ denote the set of edges adjacent to the vertices in U. We define $I(U)$ to be the join of all $I(v)$ with $v \in U$, i.e.

$$I(U) = \{t \in Z^{E(U)} \mid t_{|E(v)} \in I(v) \quad \text{for all} \quad v \in U\} \,.$$

$I(U)$ is the set of *local* solutions for U. Obviously $I(U \cup W) = I(U) \bowtie I(W)$ holds for arbitrary subsets U and W of vertices.

Definition E.2. *A (resolution)* algorithm *A for the graph G is a term A with the two-ary operation $\bowtie$ and all v in V. Its* length *$|A|$ is the number of occurrences of $\bowtie$ plus the number of occurrences of v's.*

Actually, a resolution algorithm specifies the order of all the $\bowtie$ operations starting from the $I(v)$ with $v \in V$. E.g. the term $(v_1 \bowtie v_2) \bowtie (v_3 \bowtie v_4)$ means that we first form $I(v_1) \bowtie I(v_2)$, $I(v_3) \bowtie I(v_4)$ and then the join of these two sets. There is a natural notion of subterm, the above term has the subterms $v_1 \bowtie v_2$, $v_3 \bowtie v_4$. We write $B \leq A$ if B is subterm of A.

For an arbitrary subterm B of an algorithm A let $V(B)$ denote the set of vertices occurring in B. So $I(V(B))$ is the result, or the set of local solutions, of the subterm B.

We define the logarithmic *complexity* $C_I(A)$ of an algorithm A to be the maximal logarithmic size of the result of a subterm $B \leq A$ (taking the maximum yields a simple and clean measure that differs from an average/aggregate measure only by $\log |A|$):

$$C_I(A) \;=\; \log \max_{B \leq A} \#I(V(B)) \; .$$

The logarithmic complexity roughly corresponds to a *work load* $\#Z^{C_I(A)}$. Counting only the size of the largest intermediate result is justified since the length of algorithms will be small and the costs for a join operation corresponds to the cardinality of the operands.

The *average complexity* $C(A^{\mathrm{df}})$ of algorithm A is defined to be

$$C(A^{\mathrm{df}}) \;=\; \log \; \mathrm{Exp}_I \max_{B \leq A} \#I(V(B)) \; .$$

As the distribution of I is locally uniform, the expected value Exp_I depends on df and so does $C(A^{\mathrm{df}})$. We let the *complexity* $C(G^{\mathrm{df}})$ of a computation graph G^{df} be the minimum of $C(A^{\mathrm{df}})$ over all algorithms A for G^{df}.

Getting only One Solution

In most cases, we are only interested in getting on the average *one* solution of G^{df}. If there are many solutions we can decrease the complexity by restricting the resolution process to random subsets of all the $I(v)$. Thus, for a given mapping df', with $\mathrm{df}'(v) \leq \mathrm{df}(v)$ for all v, and a given interpretation I of (G^{df}, Z), we consider a random sub-interpretation $I^{\mathrm{df}'}$ and a distribution defined by picking independently and uniformly random subsets $I^{\mathrm{df}'}(v)$ of $I(v)$ of size $\#Z^{\mathrm{df}'(v)}$. This means to consider a computation graph with df' instead of df. Note that the construction of $I^{\mathrm{df}'}$ preserves local uniformity and independence. In the following we will consider computation graphs with only one solution on the average.

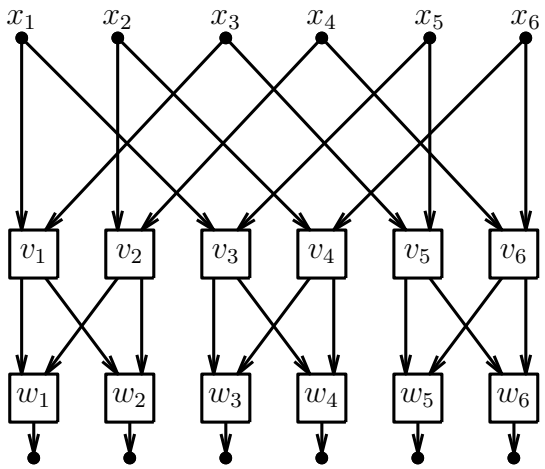


FIGURE E.1 – A powerful non-linear algorithm.

Examples of Algorithms

We demonstrate how to use exhaustive search and the birthday paradox iteratively on parts of the computation graph. Enumerating all values of a function f is formalized by the algorithm $x^1 \bowtie f^1$. Solving exhaustively $f(x) = a$ for given f, a is the intention of $(x^1 \bowtie f^1) \bowtie a^0$, where degrees of freedom $\mathrm{df}(v)$ are denoted by a superscript on v. Searching for *one* solution of $f(x, y) = a$ by picking x and y at random is the meaning of $\left(\left(x^{\frac{1}{2}} \bowtie y^{\frac{1}{2}}\right) \bowtie f^1\right) \bowtie a^0$. Here the degree of freedom of x and y have been decreased to $\frac{1}{2}$ to get one solution on the average. We can also decrease it as $((x^0 \bowtie y^1) \bowtie f^1) \bowtie a^0$ which means to fix x arbitrarily and then to search for y. Using the birthday paradox to get one solution of $f(x) = g(y)$ can be written as $\left(x^{\frac{1}{2}} \bowtie f^1\right) \bowtie \left(y^{\frac{1}{2}} \bowtie g^1\right)$. This algorithm makes two lists of $\sqrt{\#Z}$ x- and y-values and searches for matches $f(x) = g(y)$.

Linear Complexity

In [146] we only considered *linear* resolution algorithms, i.e. algorithms of the form $v_1 \bowtie (v_2 \bowtie (\ldots))$ that are characterized by the order of traversing the vertices $v_1, ..., v_n$. It is tempting to believe that linear algorithms are already the most powerful ones and that nothing can be gained from non-linear ones. The following counterexample shows this is not the case. Consider the network of figure E.1 representing the computation graph of a supposed one-way function which maps 6 inputs $x_1, \ldots, x_6$ onto 6 outputs, all 24-bits

long. The direction of the edges in figure 1 indicates the underlying network for computing the function. The problem to invert this function is defined by the following computation graph

$$V = \{x_1, \ldots, x_6, v_1, \ldots, v_6, w_1, \ldots, w_6\}$$

and

$$\mathrm{df}(x_i) = 1 \quad \mathrm{df}(v_i) = 2 \quad \mathrm{df}(w_i) = 1 \qquad i = 1, \ldots, 6.$$

It is straightforward to imagine a linear attack to invert the function with logarithmic complexity 3, that is within work load 2^{72}: enumerate x_1, x_3 and x_5 exhaustively and solve the leftmost third of the graph from x_1 and x_3 and get x_2 and x_4 as

$$A = ((((((x_1 \bowtie x_3) \bowtie v_1) \bowtie w_1) \bowtie w_2) \bowtie v_2) \bowtie x_2) \bowtie x_4.$$

Then solve the rightmost third from x_5 and get x_6 by the algorithm

$$A' = (((((A \bowtie x_5) \bowtie v_5) \bowtie w_5) \bowtie w_6) \bowtie v_6) \bowtie x_6.$$

Finally check that this yields a solution of the whole graph via

$$A'' = (((A' \bowtie v_3) \bowtie w_3) \bowtie w_4) \bowtie v_4.$$

It is easy to see that 3 is the lowest complexity for linear algorithms. On the other hand, one can solve the leftmost third from x_1 and x_3 by the algorithm A and *independently* solve the rightmost third from x_3 and x_5 by

$$B = ((((((x_3 \bowtie x_5) \bowtie v_5) \bowtie w_5) \bowtie w_6) \bowtie v_6) \bowtie x_4) \bowtie x_6$$

then join the two sets of partial solutions to get a set $A \bowtie B$ with rank 2 and finally check whether it contains a global solution

$$((((A \bowtie B) \bowtie v_3) \bowtie w_3) \bowtie w_4) \bowtie v_4.$$

This is done with a logarithmic complexity 2, that is with work load 2^{48}.

E.1.3 Approximating the Complexity

With a computation graph G^{df} we associate its *quadratic form*, which we also denote G^{df}, represented by the symmetric matrix $(G^{\mathrm{df}}_{v,w})_{v,w \in V}$, where

$$G^{\mathrm{df}}_{v,w} =_{def} \begin{cases} -\frac{1}{2} & \text{if } v \neq w \text{ and } \{v, w\} \in E \\ \mathrm{df}(v) & \text{if } v = w \\ 0 & \text{otherwise.} \end{cases}$$

The quadratic form induces a function $G^{\mathrm{df}} : \mathbf{Z}^V \longrightarrow \mathbf{Z}$ as

$$G^{\mathrm{df}}(g) \;=\; \sum_{v\in V}\sum_{w\in V} g(v)g(w)G^{\mathrm{df}}_{v,w}$$

for $g \in \mathbf{Z}^V$. We identify a subset $U \subset V$ with its characteristic function in $\{0,1\}^V$, and thus

$$G^{\mathrm{df}}(U) \;=\; \sum_{v\in U}\sum_{w\in U} G^{\mathrm{df}}_{v,w}.$$

Let $\mathrm{Int}(U) = \{\{v,w\} \in E \mid v,w \in U\}$ be the set of interior edges of U. It can easily be seen that

$$G^{\mathrm{df}}(U) \;=\; \sum_{v\in U} \mathrm{df}(v) - \#\mathrm{Int}(U) \;=\; \#E(U) + \sum_{v\in U}(\mathrm{df}(v) - d(v))$$

where the latter equality comes from $\sum\limits_{v\in U} d(v) = \#E(U) + \#\mathrm{Int}(U)$.

If $\mathrm{df}(v) = \frac{1}{2}d(v)$ holds for all vertices v then, for any subset U of vertices, $G^{\mathrm{df}}(U)$ equals to half the number of edges of the perimeter of U (i.e. the edges between U and $V-U$). We call this the *locally invertible* case as all the boxes look like permutations with the same number of inputs and outputs.

The role of the quadratic form becomes apparent in the following Lemma.

Lemma E.3. *Every subset $U \subset V$ of vertices satisfies* $\log \mathrm{Exp}_I \#I(U) = G^{\mathrm{df}}(U)$.

Proof. We note that $\mathrm{Exp}_I \#I(U) = \sum_t \Pr[t \in I(U)]$ where the sum is over all tuples t on $E(U)$. The event $\{t \in I(U)\}$ is the intersection of all the independent events $\{t_{|E(v)} \in I(v)\}$ for $v \in U$. There are $\#Z^{\#E(U)}$ many tuples on $E(U)$. Local uniformity and independence of the distribution for I yields

$$\log \mathrm{Exp}_I \#I(U) \;=\; \#E(U) + \sum_{v\in U}(\mathrm{df}(v) - d(v))$$

which, as we have already seen, equals to $G^{\mathrm{df}}(U)$. □

Theorem E.4. *Every algorithm A for G^{df} with length $|A|$ satisfies*

$$\max_{B\le A} G^{\mathrm{df}}(V(B)) \;\le\; C(A^{\mathrm{df}}) \;\le\; \log|A| \;+\; \max_{B\le A} G^{\mathrm{df}}(V(B)) \;.$$

Proof. By the definition of $C(A^{\mathrm{df}})$, and since a maximum of non-negative values is lower than their sum we have

$$\begin{aligned} C(A^{\mathrm{df}}) &= \log \mathrm{Exp}_I \max_{B\le A} \#I(V(B)) \\ &\le \log \mathrm{Exp}_I \sum_{B\le A} \#I(V(B)) \end{aligned}$$

$$\begin{aligned} &= \log \sum_{B \le A} \mathrm{Exp}_I \, \#I(V(B)) \\ &\le \log(|A| \cdot \max_{B \le A} \mathrm{Exp}_I \, \#I(V(B))) \\ &= \log|A| \; + \max_{B \le A} \log \mathrm{Exp}_I \, \#I(V(B)) \\ &= \log|A| \; + \max_{B \le A} G^{\mathrm{df}}(V(B)), \end{aligned}$$

where the last equality comes from Lemma 3.
The first inequality of the claim is straightforward by Lemma 3. □

As an immediate consequence of Theorem 4 the expression

$$C'(G^{\mathrm{df}}) := \min_A \max_{B \le A} G^{\mathrm{df}}(V(B))$$

is a close approximation to the complexity $C(G^{\mathrm{df}})$ which only depends on the quadratic form G^{df} and not on Z.

Corollary E.5. *$C'(G^{\mathrm{df}}) \le C(G^{\mathrm{df}}) \le \log|A_{opt}| + C'(G^{\mathrm{df}})$, where A_{opt} is an optimal algorithm.*

E.1.4 The Spectral Approach

We consider a locally invertible graph $G^{\frac{1}{2}d}$ so that $\mathrm{df}(v) = \frac{1}{2}d(v)$. Its quadratic form turns out to have properties similar to the Laplacian operator. In this context, lower bounds on the complexity can be proven in a similar way as in the expander graphs theory using a well-known link with the spectral values [157, 14, 13]. In this section let G be an undirected graph with n vertices and let $\lambda_1 \le \ldots \le \lambda_n$ be the eigenvalues of the quadratic form $G^{\frac{1}{2}d}$.

Lemma E.6. *If the graph G is connected then $\lambda_1 = 0$, $\lambda_2 > 0$, and every set U of c vertices satisfies $G^{\frac{1}{2}d}(U) \ge \lambda_2 \, c \, (1 - c/n)$.*

Proof. 0 is an eigenvalue since $G^{\frac{1}{2}d}(U) = 0$ holds for all connected components U of G. The fact that the quadratic form is positive and that $\lambda_2 > 0$ (if G is connected), is an easy algebra exercise. We note that λ_2 is the smallest eigenvalue of the quadratic form in the hyperplane $V^{\perp}$ orthogonal to the set V of all the vertices (*i.e.* the vector having all coordinates 1).

For an orthonormal basis of eigenvectors $v_1, \ldots, v_n$ with $v_1 = \frac{1}{\sqrt{n}}V$ we have (the dot $\cdot$ denotes the scalar product)

$$G^{\frac{1}{2}d}(U) = \sum_{i=1}^{n} \lambda_i (U \cdot v_i)^2 \ge \lambda_2 \sum_{i=2}^{n} (U \cdot v_i)^2 = \lambda_2((U \cdot U) - (U \cdot v_1)^2).$$

Now, the claim follows from $U \cdot U = c$ and $U \cdot v_1 = \frac{c}{\sqrt{n}}$. □

Then we get a lower bound:

Theorem E.7. *If the graph G is connected then $C(G^{\frac{1}{2}d}) \geq \frac{2\lambda_2}{9}n$.*

Proof. Let A be an algorithm for $G^{\frac{1}{2}d}$ such that

$$C'(G^{\frac{1}{2}d}) = \max_{B \leq A} G^{\frac{1}{2}d}(V(B)).$$

Lemma 6 yields $C'(G^{\frac{1}{2}d}) \geq \max_{B \leq A} \lambda_2 \#(V(B))(1 - \#(V(B))/n)$. Let x be an arbitrary integer between 0 and $n = \#V$. Every minimal subterm B with the property that $\#(V(B)) \geq x$ also satisfies $\#(V(B)) \leq 2x$ since it can at best be the join of two subterms which each contain $x - 1$ vertices. For such a subterm we have

$$\#(V(B))(1 - \#(V(B))/n) \geq \min_{x \leq y \leq 2x} y(1 - y/n)$$

and thus by Corollary 5

$$C(G^{\frac{1}{2}d}) \geq C'(G^{\frac{1}{2}d}) \geq \max_{0 \leq x \leq n} \min_{x \leq y \leq 2x} \lambda_2 \, y(1 - y/n) = \tfrac{2\lambda_2}{9}n.$$

The max min is obtain with $x = y = n/3$. □

E.1.5 The Symmetric Approach

Similarly, we can apply a theorem due to Babai and Szegedy [19] used in context with Cayley graphs. We reformulate it in our context and refer to [19] for the proof. Let G be an arbitrary undirected, edge-transitive graph with n vertices, let d be the harmonic mean of the degrees of the vertices and let δ be the average distance between two vertices.

Lemma E.8 (Babai-Szegedy 92). *Every set U of c vertices satisfies*

$$G^{\frac{1}{2}d}(U) \geq \tfrac{d}{2\delta}c(1 - \tfrac{c}{n}).$$

A graph is edge-transitive if every edge can be mapped onto any other one by a graph automorphism. In such a graph, the vertices can only have one or two possible degrees. This suggests to use the harmonic mean $\frac{2}{1/d_1 + 1/d_2}$ of the two degrees d_1 and d_2. A straightforward application of the proof of Theorem E.7 to Lemma 8 shows:

Theorem E.9. *Every undirected, edge-transitive graph G satisfies*

$$C(G^{\frac{1}{2}d}) \geq \frac{d}{9\delta}n.$$

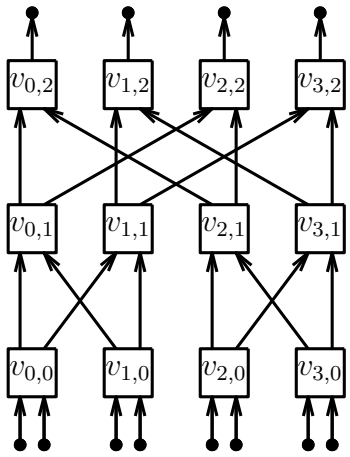


Figure E.2 – The $G_{3,2}$ compression functions family.

E.2 Parallel FFT Hashing

Two previous proposals of a cryptographic hash function based on the FFT network [141, 143] have been broken (see Baritaud-Gilbert-Girault [20] and Vaudenay [161]). Then, by a joint effort, a family of hash functions based on the same graph has been proposed in [145] and discussed in [146]. We will now prove the conjectures announced in the latter paper. Interestingly, the FFT network has independently been used by Massey for the SAFER encryption function [87].

Let $G_{k,s}$ be the graph defined by the set of vertices

$$V = \{v_{i,j}; 0 \leq i < 2^{k-1}, 0 \leq j \leq s\}$$

and the set of edges

$$E = \{\{v_{i,j}, v_{i,j+1}\}, \{v_{i,j}, v_{i \oplus 2^{j \bmod k-1}, j+1}\}; 0 \leq i < 2^{k-1}, 0 \leq j < s\}.$$

$G_{k,s}$ is roughly the graph of the FFT network for 2^k values extended to $s+1$ layers. Considering all vertices as boxes with two inputs coming from a lower layer and two outputs going to a higher layer, this corresponds to a function with 2^k inputs entering in layer 0 and 2^{k-1} outputs going out from layer s. Given two message blocks m and m' with 2^{k-1} values, we let the ith value of each block enter into vertex $v_{i,0}$ and write the first output of $v_{i,s}$ as the ith value of the output string h. The mapping $(m, m') \mapsto h$ defines a compression function. We propose to study the family of the compression functions defined

by $G_{k,s}$ and a relevant distribution of interpretations I that defines the boxes. In [145] we considered the uniform distribution on all multipermutations. The aim of the present study is to find, by a graph theoretic analysis of $G_{k,s}$, the minimal s for optimal security in the context of black-box cryptanalysis.

The one-wayness of the compression function means hardness of finding, for *given* m and h, one m' such that $(m, m') \mapsto h$. (Note: finding for given h *some* m, m' with $(m, m') \mapsto h$ is trivial as we can arbitrarily complement the "half"-output h to (h, h') and compute the inverse permutation $(h, h') \mapsto (m, m')$.) The inversion problem is defined by the computation graph $G_{k,s}$ together with

$$\mathrm{df} : v_{i,j} \mapsto \begin{cases} 1 & \text{if } j = 0 \text{ or } j = s \\ 2 & \text{otherwise.} \end{cases}$$

Here one input of the $v_{i,0}$ and one output of the $v_{i,s}$ are already known, that is $\mathrm{df} = \frac{1}{2}d = 1$ holds for the first and the last layer. The exhaustive search consists in joining all $v_{i,0}$ to guess m' and joining successively all the other vertices layer by layer. This has complexity 2^{k-1}. So, we are interested in the *ratio* $C(G_{k,s}^{\frac{1}{2}d})/2^{k-1}$.

Finally consider the length of the bit string for specifying the $(s+1)2^{k-1}$ boxes. Choosing an alphabet with cardinality q, the number of bits to encode the input is $n = 2^k \log q$ whereas the length of the description of the function (that is the interpretation) is $(s+1)2^k q^2 \log q$. The family is quite huge, but we hope to find an interesting smaller sub-family in which the following analysis will be possible too. For instance if we take the same box for all the vertices i, j and concatenate these boxes with independent random permutations along the inner edges of $G_{k,s}$ we decrease the length of the interpretation to $s2^k \log q!$ and we preserve local uniformity and independence.

E.2.1 The Upper Bounds

Theorem E.10. *For* $k - 1 \leq s \leq 2k - 2$ *we have* $C(G_{k,s}^{\frac{1}{2}d}) \leq 2^{k-2}\,(1 + 2^{2-s})$.

Thus, for $s < 2k - 2$ there is an attack faster than exhaustive search. We conjecture that this inequality is in fact an equality for $s = 2k - 2$, that is to say the exhaustive search is the best black-box attack on $G_{k,2k-2}$.

Proof. First we show that $C(G_{k,k-1}^{\frac{1}{2}d}) \leq 2^{k-2}$. For this we guess the first 2^{k-2} inputs, that is we join the first 2^{k-2} vertices $v_{i,0}$. This allows to compute half of the edges, namely all edges adjacent to $v_{i,j}$ for $i < 2^{k-2}$. Then, the degree of freedom of all $v_{i,k-1}$ becomes 0, so that we can compute the other half edges backward and solve the graph. This has complexity 2^{k-2}.

We can do similar things on $G_{k,s}$: we guess the first 2^{k-2} inputs and solve half of the vertices from layer 0 to layer $k-1$. Then, all connected subgraphs from layer $k-1$ to layer s are isomorphic to $G_{k',k'-1}$ with $k' = s-k+2$. We solve them iteratively within a complexity $2^{k'-2} = 2^{s-k}$. Having solved all of these subgraphs, we backwardly process $G_{k,s}$. The resolution has complexity $2^{k-2} + 2^{k-s}$. □

In the following sections, we show how to apply the different approaches to find lower bounds. Finally, we prove that the ratio for $s = 2k-2$ is lower bounded by a constant $\frac{2}{3}$.

E.2.2 The Lower Bounds

Use of the Spectral Approach.

We use the notation introduced in sections 1.4 and 2.

Lemma E.11. $\lambda_2\left(G_{k,s}^{\frac{1}{2}d}\right) = \begin{cases} 4\sin^2\frac{\pi}{2(2k-1)} & \text{if } k \le s < 2k-1 \\ 4\sin^2\frac{\pi}{2(s+1)} & \text{if } 2k-1 \le s. \end{cases}$

The proof is a difficult but unenlightening exercise in calculus which can be found in [163, pp. 76–80].

Proof. (sketch) We study the spectral properties of the adjacency matrix

$$M_{k,s} = \begin{pmatrix} I & -\frac{1}{2}A & 0 & \cdots & 0 & 0 \\ -\frac{1}{2}{}^tA & 2I & -\frac{1}{2}A & \cdots & 0 & 0 \\ 0 & -\frac{1}{2}{}^tA & 2I & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 2I & -\frac{1}{2}A \\ 0 & 0 & 0 & \cdots & -\frac{1}{2}{}^tA & I \end{pmatrix}$$

where I is the identity matrix and

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & 1 \\ 1 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & 1 \end{pmatrix}.$$

We let f_α denote the Hadamard vector basis: for any boolean vector α with $k-1$ coordinates, f_α is a real vector with 2^{k-1} entries. Each entry index corresponds to a boolean vector i with $k-1$ coordinates and f_α is defined by $(f_\alpha)_i = (-1)^{\alpha \cdot i}$ where $\cdot$ is the dot product. In the basis of all block vectors $(0, \ldots, 0, f_\alpha, 0, \ldots, 0)$, the matrix $M_{k,s}$ is block-diagonal with $(s+1)m \times (s+1)m$-blocks of the form

$$B_s(J) = \begin{pmatrix} I & -J & 0 & \cdots & 0 & 0 \\ -{}^tJ & 2I & -J & \cdots & 0 & 0 \\ 0 & -{}^tJ & 2I & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 2I & -J \\ 0 & 0 & 0 & \cdots & -{}^tJ & I \end{pmatrix}$$

where J is a Jordan $m \times m$-block and ${}^tJ$ its transpose. Studying the spectral properties of the $B_s(J)$ is a technical algebra exercise. □

Then Theorem E.7 implies:

Corollary E.12. $C\left(G_{k,s}^{\frac{1}{2}d}\right) \geq 2^{k-1} \begin{cases} \frac{8(s+1)}{9} \sin^2 \frac{\pi}{2(2k-1)} & \text{if } k \leq s < 2k-1 \\ \frac{8(s+1)}{9} \sin^2 \frac{\pi}{2(s+1)} & \text{if } 2k-1 \leq s. \end{cases}$

This suggests to use $s = 2k-1$ to get optimal security. The lower bound of the ratio is here equivalent to $\frac{2\pi^2}{9s}$.

Use of the Symmetric Approach.

Though $G_{k,s}$ is not edge-transitive, it is possible to use the symmetric approach for the case $s = k$. If we contract layers 0 and 1 following the rule that adjacent edges are merged, we get the same result (*i.e.* isomorphic) as if we add symmetrical edges between the first and the last edges of $G_{k-1,2k-5}$. The obtained graph $G'_{k-1,2k-5}$ turns out to be edge-transitive for $k \geq 3$. This graph has degree $d = 4$.

Lemma E.13. *The average distance of $G'_{k-1,2k-5}$ is asymptotically $\frac{3}{2}(k-2)$.*

Proof. (sketch) By studying the distance between two vertices, we obtain that the average distance is

$$\delta = \frac{3}{2}(k-2) + \frac{C_{k-2}}{k-2} - \frac{2}{k-2} \sum_{i=1}^{k-2} C_i$$

where C_i is the average length of the longest all-zero subsequence of a random boolean sequences of length i. Then we prove that $C_i \leq 1 + \log_2 i$. So, $\delta \sim \frac{3}{2}(k-2)$. The complete proof can be found in Vaudenay [163, pp. 74–76]. □

We let $C_{\mathrm{lin}}(G_{k,k}^{\frac{1}{2}d})$ denotes the linear complexity of $G_{k,k}^{\frac{1}{2}d}$, that is

$$C_{\mathrm{lin}}(G_{k,k}^{\frac{1}{2}d}) = \min_{A} C(A^{df})$$

over all linear algorithms A. The last Lemma allows to prove a technical corollary we mention for completeness.

Corollary E.14. *For $k \geq 3$ we obtain* $C_{\mathrm{lin}}\left(G_{k,k}^{\frac{1}{2}d}\right) \geq \frac{2^{k-1}}{3}(1 + o(1))$.

This establishes the lower bound $\frac{1}{3}$ for the ratio. Unfortunately, we could not prove a similar result for the general complexity following this approach.

Proof. (sketch) Any linear algorithm A can be rewritten, without increasing its complexity, as an algorithm such that for any subterm B which involves a vertex $v_{i,j}$ for $j = 0$, 1, $s-1$ or s, every of the other vertices which are merged with $v_{i,j}$ either already occurs in B or will all be joined immediately hereafter. In such an algorithm, there is at least one out of four consecutive subterms B which has all of its merged *classes complete*, i.e. closed with respect to merging. Let B' be the merged image of such a B in $G'_{k-1,2k-5}$. The completeness property of B implies $G_{k,k}(B) = G'_{k-1,2k-5}(B')$. Thus, using the Babai-Szegedy theorem together with the previous lemma, we obtain

$$G_{k,k}^{\frac{1}{2}d}(V(B)) \geq \frac{4}{3(k-2)}\#(V(B'))\left(1 - \frac{\#(V(B'))}{(2k-4)2^{k-2}}\right).$$

We observe that

$$\#(V(B)) - 3 \cdot 2^{k-1} \leq \#(V(B')) \leq \#(V(B)).$$

and that at least one out of four consecutive B satisfies the *completeness* property, hence the result. □

Use of the Flow Approach.

In the particular case of the graph $G_{k,2k-2}$, there is an independent method, dedicated to this graph and based on the min-cut max-flow theorem.

Lemma E.15. *Let V_0 and V_s be respectively the first and the last layer of $G_{k,2k-2}$. For every function r from $V_0 \cup V_s$ to $\mathbf{Z}$ such that $|r(v)| \leq 2$ and $\sum_v r(v) = 0$ there exists a flow f with source r, that is to say a function from the set $\bar{E}$, the set of directed edges, to $\mathbf{Z}$ such that for all v and w:*

1. $f(v, w) = -f(w, v)$,

2. $|f(v,w)| \leq 1$,
3. $\sum_u f(u,v) = \begin{cases} r(v) & \text{if } v \in V_0 \cup V_s \\ 0 & \text{otherwise.} \end{cases}$

Proof. Let $\{v_{i,j}, v_{i',j+1}\}$ be an edge with $j+1 \leq k-1$. We define

$$f(v_{i,j}, v_{i',j+1}) = \frac{1}{2} \operatorname*{Mean}_{v_{i'',0}} r(v_{i'',0})$$

where the arithmetic mean is taken over all vertices $v_{i'',0}$ such that there exists a straight path in $G_{k,2k-2}$ from $v_{i'',0}$ to $v_{i,j}$. $f(v_{i,j}, v_{i',j+1})$ is thus equal to half the mean of all $r(v_{i'',0})$ the incoming flow in $v_{i,j}$ from previous layers which is equally spread into its two outgoing edges. Hence, the incoming flow in all $v_{i,k-1}$ will be a constant. Defining $f(v_{i',j+1}, v_{i,j}) = -f(v_{i,j}, v_{i',j+1})$, it is easy to show that the above three conditions are satisfied for all edges before the layer $k-1$.

Similarly, for $j \geq k-1$, we define $f(v_{i,j}, v_{i',j+1})$ to be half the incoming flow in $v_{i',j+1}$ from the next layers starting at V_s. The conditions are satisfied for all edges after the layer $k-1$. The flow coming from all the upper layers to the layer $k-1$ is also equally spread into all the vertices. Then, due to the condition that the sum of all $r(v)$ is 0, the third condition is also satisfied in the layer $k-1$. □

Using the previous lemma we show:

Lemma E.16. *For any set U in $G_{k,2k-2}$, if $c = \#(U \cap (V_0 \cup V_s))$ where V_0 is the first layer and V_s is the last one, we have $G^{\frac{1}{2}d}_{k,2k-2}(U) \geq 2^{k-1} - |2^{k-1} - c|$.*

Proof. We note that $G^{\frac{1}{2}d}_{k,2k-2}(U) = G^{\frac{1}{2}d}_{k,2k-2}(V-U)$ (this comes from the fact that V is in the kernel of the quadratic form). Thus, possibly replacing U by $V-U$, we can assume $c \leq 2^{k-1}$. Now, for $v \in V_0 \cup V_s$, we can define $r(v)$ to be 2 if $v \in U$ and $r(v) = -\frac{2c}{2^{k}-c}$ otherwise. Since r satisfies the conditions of the lemma, there exists a flow with source r, hence with capacity $2c$. We note that $2G^{\frac{1}{2}d}_{k,2k-2}(U)$ is equal to the cardinality of the border of A, which is a cut for the flow. Hence, the min-cut max-flow theorem says $2G^{\frac{1}{2}d}_{k,2k-2}(U) \geq 2c$. □

Corollary E.17. $C(G^{\frac{1}{2}d}_{k,2k-2}) \geq \frac{2}{3}\, 2^{k-1}$.

Proof. In a similar way as in the proof of the theorem E.7, we get

$$C(G^{\frac{1}{2}d}_{k,2k-2}) \geq \max_{0 \leq x \leq 2^k} \min_{x \leq y \leq 2x} \left(2^{k-1} - |2^{k-1} - x|\right)$$

which is equal to $\frac{2}{3}2^{k-1}$. □

A similar method applied on $G_{k,k-1}$ (basically, in taking only V_0 into account in the source of the flow) enables to prove:

Corollary E.18. $C(G_{k,k-1}^{\frac{1}{2}d}) \geq \frac{1}{3}\,2^{k-1}$.

This confirms the partial result obtained by the symmetric approach.

These results establish a constant ratio between the upper bounds and the lower bounds. Actually, the same method applied to linear algorithms shows that $C_{\text{lin}}(G_{k,s}) = 2^{k-1}$, which proves the conjecture in [146]. We conjecture that the upper bounds are the real complexities also in the general case for $s = 2k - 2$. This means that one has to choose $s = 2k - 2$ to get the optimal security for the $G_{k,s}$ compression function family.

E.3 Possible Extensions and Conclusion

The analysis on cryptographic primitives proposed here can be extended in a more general context. To allow several edge domains to exist together in the same primitive, we can add the notion of edge degree of freedom in the definition of the computation graphs. This would be the logarithm (in any basis) of the cardinality of the domain. We mention that all the results still hold if we replace $d(v)$ by the sum of all the adjacent edge degrees. We can also allow the value of an edge to be involved in more than two different vertices replacing the notion of graph by the notion of hypergraph.

We have proposed a new framework for the study of the security of cryptographic primitives defined as a computation graph. We showed that the complexity of resolving a computation graph is related to the local expansion properties of the graph. This theory enables one to prove the one-wayness of a family of compression functions with respect to the black-box attacks. It can be applied to the compression functions based on the FFT network. It turns out that the function is the most secure possible (in context with the black-box cryptanalysis) for a doubled FFT network, that is for $s = 2k - 2$.

Acknowledgment

We thank László Babai and Jacques Stern for helpful discussions. We also thank the anonymous referees for extensive and useful remarks.

Annexe F

Provable Security for Block Ciphers by Decorrelation

[*Cet article de* Serge Vaudenay *a été soumis au* Journal of Cryptology *[174]. Il rassemble des résultats présentés aux colloques Stacs 98 et Sac 98 [173, 176].*]

Abstract. In this paper we investigate a new way for protecting block ciphers against classes of attacks (including differential and linear cryptanalysis) which is based on the notion of decorrelation distance which is fairly connected to Carter-Wegman's universal hash functions paradigm. This defines a simple and friendly combinatorial measurement which enables to quantify the security. We show that we can mix provable protections and heuristic protections. We finally propose two new block cipher families we call COCONUT and PEANUT, which implement these ideas and achieve quite reasonable performances for real-life applications.

Before the second world war, security of encryption used to be based on the secrecy of the algorithm. Mass telecommunication and computer science networking however pushed the development of public algorithms with secret keys. The most important research result on encryption was found for the application to the telegraph by Shannon in the Bell Laboratories in 1949 [150]. It proved the unconditional security of the Vernam's Cipher which had been published in 1926 [181]. Although quite expensive to implement for networking (because the sender and the receiver need to be synchronized, and it needs quite cumbersome huge keys), this cipher was used in the Red Telephone between Moscow and Washington D.C. during the cold war. Shannon's result also proves that unconditional security cannot be achieved in a better

(*i.e.* cheaper) way. For this reason, empiric security seemed to be the only efficient possibility, and all secret key block ciphers which have been publicly developed were considered to be secure until some researcher published an attack on it. Therefore research mostly grew like a ball game between the designers team and the analysts team and treatment on the general security of block ciphers has hardly been done.

In adopting the *Data Encryption Standard* (DES) [2] in the late 70's, the U.S. Government classified the development arguments. Attacking DES was thus quite challenging, and this paradoxically boosted research on block ciphers. Real advances on the security on block ciphers have been made in the early 90's.

One of the most important result has been obtained by Biham and Shamir in performing a *differential cryptanalysis* on DES [27, 28, 29, 30]. The best version of this attack can recover a secret key with a simple 2^{47}-chosen plaintext attack[1]. Although this attack is heuristic, experiments confirmed the results.

Biham and Shamir's attack was based on statistical cryptanalysis idea which have also been used by Gilbert and Chassé against another cipher [54, 53]. Those ideas inspired Matsui who developed a *linear cryptanalysis* on DES [89, 91]. This heuristic attack, which has been implemented, can recover the key with a 2^{43}-known plaintext attack. Since then, many researchers tried to generalize and improve these attacks (see for instance [81, 79, 63, 74, 163, 71, 107, 165]), but the general ideas was quite the same.

The basic idea of differential cryptanalysis is to use properties like "if x and x' are two plaintext blocks such that $x' = x + a$, then it is likely that $C(x') = C(x)+b$". Then the attack is an iterated two-chosen plaintexts attack which consists in getting the encrypted values of two random plaintexts which verify $x' = x + a$ until a special event like $C(x') = C(x) + b$ occurs. Similarly, the linear cryptanalysis consists in using the probability $\Pr[C(x) \in H_2 / x \in H_1]$ for two given hyperplanes H_1 and H_2. With the GF(2)-vector space structure, hyperplanes are half-spaces, and this probability shall be close to $1/2$. Linear cryptanalysis uses the distance of this probability to $1/2$ when it is large enough. More precisely, linear cryptanalysis is an incremental one-known plaintext attack where we simply measure the correlation between the events $[x \in H_1]$ and $[C(x) \in H_2]$.

Instead of keeping on breaking and proposing new encryption functions, some researchers tried to focus on the way to protect ciphers against some classes of attacks. Nyberg first formalized the notion of strength against

[1] So far, the best known attack was an improvement of exhaustive search which requires on average 2^{54} DES computations.

differential cryptanalysis [110], and similarly, Chabaud and Vaudenay formalized the notion of strength against linear cryptanalysis [35]. With this approach we can study how to make internal computation boxes resistant against both attacks. This can be used in a heuristic way by usual active s-boxes counting tricks (*e.g.*, see [63, 64]). This has also been used to provide provable security against both attacks by Nyberg and Knudsen [115], but in an unsatisfactory way which introduce some algebraic properties which lead to other attacks as shown by Jakobsen and Knudsen [67].

In this presentation, we introduce a new way to protect block ciphers against various kind of attacks. This approach is based on the notion of universal functions introduced by Carter and Wegman [34, 183] for the purpose of authentication. Protecting block ciphers is so cheap that we call NUT (as for "n-Universal Transformation") the added operations which provide this security. We finally describe two cipher families we call COCONUT (as for "Cipher Organized with Cute Operations and NUT") and PEANUT (as for "Pretty Encryption Algorithm with NUT") and offer two definite examples as a cryptanalysis challenge.

The paper is organized as follows. First we give some definitions on decorrelation distance (Section F.1) and basic constructions (Section F.2). Then we state Shannon's perfect secrecy notion in term of decorrelation distance (Section F.3). We show how to express security results in the Luby-Rackoff's security model (Section F.4). Then we compute how much Feistel Ciphers can be decorrelated (Section F.5). We prove how pairwise decorrelation can protect a cipher against differential cryptanalysis and linear cryptanalysis (Sections F.6 and F.7). We generalize those results with the notion of "attacks of order d" (Section F.8). Finally, we define the COCONUT and PEANUT families (Sections F.9 and F.10).

F.1 Decorrelation Distance

For a treatment on block cipher, we consider ciphers as random permutations C on the message-block space $\mathcal{M}$. (Here the randomness comes from the random choice of the secret key.) In most of practical cases, we have $\mathcal{M} = \{0,1\}^m$.

We first give formal definitions of the notion of decorrelation distance which plays a crucial role in our treatment.

Definition F.1. *Given a random function F from a given set $\mathcal{M}_1$ to a given set $\mathcal{M}_2$ and an integer d, we define the* d-wise distribution matrix $[F]^d$ *of F as a $\mathcal{M}_1^d \times \mathcal{M}_2^d$-matrix where the (x,y)-entry of $[F]^d$ corresponding to the*

multi-points $x = (x_1, \ldots, x_d) \in \mathcal{M}_1^d$ *and* $y = (y_1, \ldots, y_d) \in \mathcal{M}_2^d$ *is defined as the probability that we have* $F(x_i) = y_i$ *for* $i = 1, \ldots, d$.

Basically, each row of the d-wise distribution matrix corresponds to the distribution of the d-tuple $(F(x_1), \ldots, F(x_d))$ where $(x_1, \ldots, x_d)$ corresponds to the index of the row.

Definition F.2. *Given two random functions* F *and* G *from a given set* $\mathcal{M}_1$ *to a given set* $\mathcal{M}_2$*, an integer* d *and a distance* D *over the vector space* $\mathbf{R}^{\mathcal{M}_1^d \times \mathcal{M}_2^d}$*, we call* $D([F]^d, [G]^d)$ *the* d-wise decorrelation D-distance *between* F *and* G.

A decorrelation distance of zero means that for any multi-point

$$x = (x_1, \ldots, x_d)$$

the multi-points $(F(x_1), \ldots, F(x_d))$ and $(G(x_1), \ldots, G(x_d))$ have the same distribution, so that F and G have the same *decorrelation*.

Actually, we do not need a distance over the whole matrix set, but only on some sub-algebra. We distinguish the decorrelation of *functions* from the decorrelation of *permutations* (or *ciphers*). Distribution matrices A of functions (as well as other matrices in the sub-algebra they span) are such that if $x_i = x_j$ for some indices i and j in any multi-point x, then for all multi-point y such that $y_i \neq y_j$ we have $A_{x,y} = 0$ (because if $x_i = x_j$ then $F(x_i) = F(x_j)$). Additional properties hold for permutations. Thus, we indeed need distance over the corresponding sub-algebra.

It is also important to study the decorrelation distance of a given random function F to a reference random function. Random functions F are compared to uniformly distributed random functions (that we call *perfect random functions*) which will be denoted F^*. We say that the decorrelation of the random function F^* is *perfect*. Similarly, random permutations C are compared to a uniformly distributed permutations C^* (which will be called *perfect cipher*), and the *decorrelation of the cipher* C^* is *perfect*. Any random function (resp. cipher) with a d-wise decorrelation distance of zero to the perfect random function (resp. perfect cipher) will be said to have a *perfect decorrelation*.

For instance, let say that F is a random function from $\mathcal{M}_1$ to $\mathcal{M}_2$. Saying that F has a perfect 1-wise decorrelation means that for any x_1 the distribution of $F(x_1)$ is uniform. Saying that the *function* F has a perfect 2-wise decorrelation means that for any $x_1 \neq x_2$ the random variables $F(x_1)$ and $F(x_2)$ are uniformly distributed and independent. Saying that a *cipher* C on $\mathcal{M}$ has a perfect 2-wise decorrelation means that for any $x_1 \neq x_2$, the random variable $(C(x_1), C(x_2))$ is uniformly distributed among all the (y_1, y_2) pairs such that $y_1 \neq y_2$.

Definition F.3. *Let F (resp. C) be a random function from $\mathcal{M}_1$ to $\mathcal{M}_2$ (resp. a random permutation over $\mathcal{M}$). Let D be a distance over the algebra spanned by the d-wise distribution matrices of random functions (resp. of random permutations). We call* d-wise decorrelation D-bias *and we denote* ${}^{f}\mathrm{Dec}^d_D(F)$ *(resp.* $\mathrm{Dec}^d_D(C)$*) the quantity $D([F]^d, [F^*]^d)$ (resp. $D([C]^d, [C^*]^d)$) where F^* (resp. C^*) is uniformly distributed.*

We note that this notion is fairly similar to the notion of universal functions which was been introduced by Carter and Wegman [34, 183]. More precisely, we recall that a random function F from $\mathcal{M}_2$ to $\mathcal{M}_2$ is ϵ-almost strongly universal$_d$ if for any pairwise different $(x_1, \ldots, x_d)$ and any $(y_1, \ldots, y_d)$ we have

$$\Pr[F(x_i) = y_i; i = 1, \ldots, d] \leq \frac{1}{\#\mathcal{B}^d} + \epsilon.$$

If we define $||A||_\infty = \max_{x,y} |A_{x,y}|$, if the function F has a d-wise decorrelation $||.||_\infty$-bias of ϵ, then it is ϵ-almost strongly universal$_d$. The converse is true when $\epsilon \geq \frac{1}{\#\mathcal{B}^d}$. Although the notion is fairly similar, we will use our formalism which is adapted to our context.

For the purpose of our treatment, we define the L_2 norm, the infinity weighted norm N_∞ and the L_∞-associated matrix norm $|||.|||_\infty$ on $\mathbf{R}^{\mathcal{M}_1^d \times \mathcal{M}_2^d}$ by:

$$||A||_2 = \sqrt{\sum_{x,y} (A_{x,y})^2} \tag{F.1}$$

$$N_\infty(A) = \max_{x,y} \frac{|A_{x,y}|}{\Pr[x \overset{C^*}{\mapsto} y]} \tag{F.2}$$

$$|||A|||_\infty = \max_x \sum_y |A_{x,y}| \tag{F.3}$$

where C^* is the Perfect Cipher. We note that the N_∞ can only be defined on the sub-algebra spanned by distribution matrices of ciphers *i.e.* with the convention that $0/0 = 0$.

We recall that the $||.||_2$ and $|||.|||_\infty$ norms are matrix norms, *i.e.* $||A \times B|| \leq ||A||.||B||$. Moreover, N_∞ has a similar property in its sub-algebra of definition. Multiplicativity of the decorrelation distance to C^* is very useful when we consider product ciphers. Concretely, if C_1 and C_2 are two independent ciphers, then

$$\mathrm{Dec}^d(C_1 \circ C_2) \leq \mathrm{Dec}^d(C_1).\mathrm{Dec}^d(C_2)$$

for any matrix norm. (This comes from $[C_1 \circ C_2]^d = [C_2]^d \times [C_1]^d$ and $[C_i]^d \times [C^*]^d = [C^*]^d$.) This property makes the decorrelation bias a multiplicative combinatorial measurement whenever the norm is a matrix norm.

F.2 Basic Constructions

Perfect 1-wise decorrelation is easy to achieve when the message-block space $\mathcal{M}$ is given a group structure. For instance we can use $C(x) = x+K$ where K has a uniform distribution on $\mathcal{M}$, which is exactly Vernam's Cipher ([181]).

We can construct perfect pairwise decorrelated ciphers on a field structure $\mathcal{M}$ by $C(x) = a.x + b$ where $K = (a, b)$ is uniform in $\mathcal{M}^* \times \mathcal{M}$. This requires to consider the special case $a = 0$ when generating K. On the standard space $\mathcal{M} = \{0,1\}^m$, it also requires to implement arithmetic on the finite field $\mathrm{GF}(2^m)$, which may lead to poor encryption rate on software. As an example we can mention the COCONUT Ciphers (see Section F.9).

A similar way to construct (almost) perfect 3-wise decorrelated ciphers on a field structure $\mathcal{M}$ is by $C(x) = a + b/(x + c)$ where $K = (a, b, c)$ with $b \neq 0$. (By convention we set $1/0 = 0$.)

Perfect decorrelated ciphers of higher orders require dedicated structure. We can for instance use Dickson's Polynomials.

An alternate way consists of using Feistel Ciphers with decorrelated functions [51]. Given a set $\mathcal{M} = \mathcal{M}_0^2$ where $\mathcal{M}_0$ has a group structure and given r random functions $F_1, \ldots, F_r$ on $\mathcal{M}_0$ we denote $C = \Psi(F_1, \ldots, F_r)$ the cipher defined by $C(x^l, x^r) = (y^l, y^r)$ where we iteratively compute a sequence (x_i^l, x_i^r) such that

$$\begin{aligned} x_0^l = x^l \quad &\text{and} \quad x_0^r = x^r \\ x_i^l = x_{i-1}^r \quad &\text{and} \quad x_i^r = x_{i-1}^l + F_i(x_{i-1}^r) \\ y^l = x_r^r \quad &\text{and} \quad y^r = x_r^l. \end{aligned}$$

(Note that the final exchange between the two halves is canceled here.) In most of the constructions, $\mathcal{M}_0$ is the group $\mathbf{Z}_2{}^{\frac{m}{2}}$, so the addition is the bitwise exclusive *or*.

If $\mathcal{M}_0$ has a field structure, we can use perfect d-wise decorrelated F_i functions by $F_i(x) = a_d.x^{d-1} + \ldots + a_2.x + a_1$ where $(a_1, \ldots, a_d)$ is uniformly distributed on $\mathcal{M}_0^d$.

Decorrelation of Feistel Ciphers depends on the decorrelation of all F_i functions. It will be studied in Section F.5.

F.3 Shannon's Unconditional Security

In this section, we consider perfect decorrelation.

Intuitively, if C has a perfect 1-wise decorrelation, the encryption $C(x_1)$ contains no information on the plaintext-block x_1, so the cipher C is secure

if we use it only once (as one-time pad [181]). This corresponds to Shannon's perfect secrecy theory [150]. Similarly, if C has a perfect d-wise decorrelation, it is unconditionally secure if we use it only d times (on different plaintexts) as the following theorem shows.

Theorem F.4. *Let C be a cipher with a perfect d-wise decorrelation. For any $x_1, \ldots, x_{d-1}$, if X is a random variable such that $X \neq x_i$, then*

$$H(X/C(x_1), \ldots, C(x_{d-1}), C(X)) = H(X)$$

where H denotes Shannon's entropy of random variables.

This means that if an adversary knows $d-1$ pairs $(x_i, C(x_i))$, for any y_d which is different from all $C(x_i)$'s, his knowledge of $C^{-1}(y_d)$ is exactly that it is different from all x_i's. We recall that by definition we have $H(X/Y) = H(X,Y) - H(Y)$ and

$$H(X) = -\sum_x \Pr[X = x] \log_2 \Pr[X = x]$$

with the convention that $0 \log_2 0 = 0$.

Proof. From definitions, straightforward computations shows that for any random variable X we have

$$H(X/C(x_1), \ldots, C(x_{d-1}), C(X)) = H(X) + p \log_2 p$$

where $p = \Pr[X \neq x_i; i = 1, \ldots, d-1]$. Hence for any random variable such that $\Pr[X = x_i] = 0$ the property holds. □

F.4 Security in the Luby-Rackoff Model

To illustrate the power of the notion of decorrelation, let us first measure the unconditional security. In the Luby-Rackoff model, an attacker is an infinitely powerful Turing machine $\mathcal{A}^{\mathcal{O}}$ which has access to an oracle $\mathcal{O}$ whose aim is to distinguish a cipher C from the Perfect Cipher C^* by querying the oracle which implements either cipher, and with a limited number d of inputs (see [86]). The oracle $\mathcal{O}$ either implements C or C^*, and that the attacker must finally answer 0 ("reject") or 1 ("accept"). We measure the ability to distinguish C from C^* by the advantage $\mathrm{Adv}_{\mathcal{A}}(C) = |p - p^*|$ where p (resp. p^*) is the probability of answering 1 if $\mathcal{O}$ implements C (resp. C^*). A general distinguisher is illustrated on Fig. F.1. We have the following theorem.

Input: an oracle which implements a permutation c

1. calculate a message X_1 and get $Y_1 = c(X_1)$
2. calculate a message X_2 and get $Y_2 = c(X_2)$
3. ...
4. calculate a message X_d and get $Y_d = c(X_d)$
5. depending on $X = (X_1, \ldots, X_d)$ and $Y = (Y_1, \ldots, Y_d)$, output 0 or 1

FIGURE F.1 – A General d-Limited Distinguisher.

Theorem F.5. *Let d be an integer and C be a cipher. For any general d-limited distinguisher (depicted on Fig. F.1), we have*

$$\mathrm{Adv}_{\mathrm{Fig.F.1}}(C) \leq \mathrm{Dec}^d_{N_\infty}(C)$$

where the N_∞ norm is defined by Equation (F.2).

In particular, we have unconditional security when the decorrelation is perfect and we still have a proven quantified security when the decorrelation is small.

Proof. Each execution of the attack with an oracle which implements C is characterized by a random tape ω and the successive answers $y_1, \ldots, y_d$ of the queries which we denote $x_1, \ldots, x_d$ respectively. More precisely, x_1 depends on ω, x_2 depends on ω and y_1 and so on. The answer thus depends on $(\omega, y_1, \ldots, y_d)$. Let A be the set of all $(\omega, y_1, \ldots, y_d)$ such that the output of the distinguisher is 1 and let $\epsilon = \mathrm{Dec}^d_{N_\infty}(C)$. We have

$$\begin{aligned} p &= \sum_{(\omega, y_1, \ldots, y_d) \in A} \Pr[\omega] \Pr[C(x_i(\omega, y_1, \ldots, y_{i-1})) = y_i; i = 1, \ldots, d] \\ &\leq (1+\epsilon) \sum_{(\omega, y_1, \ldots, y_d) \in A} \Pr[\omega] \Pr[C^*(x_i(\ldots)) = y_i; i = 1, \ldots, d] \\ &\leq (1+\epsilon) p^* \end{aligned}$$

so we have $p - p^* \leq \epsilon$ for any attacker. We can apply this result to the attacker which produces the opposite output to complete the proof. □

Here is a more intuitive meaning of Theorem F.5 which is interesting when we use encryption as a message authentication code.

Corollary F.6. *Let d be an integer and C be a cipher on a space of size M. For any chosen plaintext attack which can query up to $d-1$ $C(x_i)$ values and which issues a (x_d, y_d) pair with $x_d \neq x_i$ $(i = 1, \ldots, d-1)$, the probability that $y_d = C(x_d)$ is at most $\frac{1}{M} + \mathrm{Dec}^d_{N_\infty}(C)$.*

Input: an oracle which implements a permutation c

1. calculate some messages $X = (X_1, \ldots, X_d)$
2. get $Y = (c(X_1), \ldots, c(X_d))$
3. depending on X and Y, output 0 or 1

Figure F.2 – A d-Limited Non-Adaptive Distinguisher.

Proof. Such an attack can be transformed into a d-limited distinguisher: the distinguisher first simulate the attack by obtaining $d-1$ pairs from the oracle and obtain a (x_d, y_d) pair. It then queries the oracle with x_d and output 1 if, and only if $y_d = C(x_d)$. From the fact that the advantage is at most ϵ we obtain the result. □

Here is a more precise theorem in the non adaptive case. We call a distinguisher "non adaptive" if no x_i queried to the oracle depends on some previous answers y_j (see Fig. F.2).

Theorem F.7. *Let d be an integer and C be a cipher. The best d-limited non-adaptive distinguisher (depicted on Fig. F.2) for C is such that*

$$\mathrm{Adv}_{\text{Fig.F.2}}(C) = \frac{1}{2}\mathrm{Dec}^{d}_{|||.|||_\infty}(C)$$

where the $|||.|||_\infty$ norm is defined by Equation (F.3).

Proof. For those attacks, with the notations of Theorem F.5, we have

$$p = \sum_x \Pr[x] \sum_y 1_{(x,y)\in A} \Pr\left[x \overset{C}{\mapsto} y\right]$$

(where 1_P is defined to be 1 if predicate P is true and 0 otherwise) thus, for the best attack, we have

$$|p - p^*| = \max_{\substack{x \mapsto \Pr[x] \\ A}} \left| \sum_x \Pr[x] \sum_y 1_{(x,y)\in A} \left(\Pr\left[x \overset{C}{\mapsto} y\right] - \Pr\left[x \overset{C^*}{\mapsto} y\right] \right) \right|.$$

We can easily see that this maximum is obtained when $x \mapsto \Pr[x]$ is a Dirac distribution on a multi-point $x = x^0$ and A includes all y's such that $\Pr\left[x_0 \overset{C}{\mapsto} y\right] - \Pr\left[x_0 \overset{C^*}{\mapsto} y\right]$ has the same sign, which gives the result. □

Here is a more intuitive consequence analog to Corollary F.6.

Corollary F.8. *Let d be an integer and C be a cipher on a space of size M. For any chosen plaintext attack which aims to compute $C(x_d)$ for a given x_d and which only can query for $d-1$ chosen values $C(x_i)$ for $x_i \neq x_d$ $(i = 1, \ldots, d-1)$ in a non-adaptive way, the probability of success is at most $\frac{1}{M} + \mathrm{Dec}^d_{|||.|||_\infty}(C)$.*

F.5 Decorrelation of Feistel Ciphers

In this section, we assume that $\mathcal{M} = {\mathcal{M}_0}^2$ where $\mathcal{M}_0$ is a group. Thus we can consider Feistel Ciphers on $\mathcal{M}$.

Theorem F.7 can be used in a non-natural way. For instance, let us recall the following theorem.

Theorem F.9 (Luby-Rackoff [86]). *Let F_1, F_2, F_3 be three independent uniform random functions on $\mathcal{M}_0$ and d be an integer. For any distinguishing attacker $\mathcal{A}$ against $\Psi(F_1, F_2, F_3)$ on $\mathcal{M} = {\mathcal{M}_0}^2$ which is limited to d queries, we have*

$$\mathrm{Adv}_{\mathcal{A}}(\Psi(F_1, F_2, F_3)) \leq \frac{d^2}{\sqrt{\#\mathcal{M}}}.$$

Thus from Theorem F.7 we have

$$\mathrm{Dec}^d_{|||.|||_\infty}(\Psi(F_1, F_2, F_3)) \leq 2\frac{d^2}{\sqrt{\#\mathcal{M}}}.$$

For completeness, we also mention some improvements to the previous theorem due to Patarin [118, 121].

Theorem F.10 (Patarin [121]). *Let $F_1, \ldots, F_6$ be six independent uniform random functions on $\mathcal{M}_0$ and d be an integer. For any distinguishing attacker against $\Psi(F_1, \ldots, F_6)$ on $\mathcal{M} = {\mathcal{M}_0}^2$ which is limited to d queries $\mathcal{A}$, we have*

$$\mathrm{Adv}_{\mathcal{A}}(\Psi(F_1, \ldots, F_6)) \leq \frac{37d^4}{(\#\mathcal{M})^{\frac{3}{2}}} + \frac{6d^2}{\#\mathcal{M}}.$$

So, as Theorem F.9 guarantees the security of a three-round Feistel Cipher for $d = \Omega\left((\#\mathcal{M})^{\frac{1}{4}}\right)$, this one guarantees the security for $d = \Omega\left((\#\mathcal{M})^{\frac{3}{8}}\right)$.

The decorrelation $|||.|||_\infty$-bias of Feistel Ciphers can be estimated with the following lemma.

Lemma F.11. *Let $F_1, \ldots, F_r$ (resp. $R_1, \ldots, R_r$) be r independent random functions on $\mathcal{M}_0$ such that $|||[F_i]^d - [R_i]^d|||_\infty \leq \epsilon_i$ $(i = 1, \ldots, r)$. We have*

$$|||[\Psi(F_1, \ldots, F_r)]^d - [\Psi(R_1, \ldots, R_r)]^d|||_\infty \leq (1+\epsilon_1)\ldots(1+\epsilon_r) - 1.$$

Proof. Let u^i denotes the input of F_i (resp. R_i) in $\Psi(F_1, \ldots, F_r)$ (resp. $\Psi(R_1, \ldots, R_r)$). We thus let (u^0, u^1) denotes the input of the ciphers, and (u^{r+1}, u^r) denotes the output. Here, all u^is are multi-points, *i.e.*

$$u^i = (u^i_1, \ldots, u^i_d).$$

We have

$$\begin{aligned}
&\Pr_{F_1,\ldots,F_r}[u^0u^1 \mapsto u^{r+1}u^r] - \Pr_{R_1,\ldots,R_r}[u^0u^1 \mapsto u^{r+1}u^r] \\
&= \sum_{u^2,\ldots,u^{r-1}} \left(\prod_{i=1}^{r} \Pr_{F_i}[u^i \mapsto u^{i+1} - u^{i-1}] - \prod_{i=1}^{r} \Pr_{R_i}[u^i \mapsto u^{i+1} - u^{i-1}] \right) \\
&= \sum_{u^2,\ldots,u^{r-1}} \sum_{\substack{I \subseteq \{1,\ldots,r\} \\ I \neq \emptyset}} \left(\prod_{i \in I} (\Pr_{F_i} - \Pr_{R_i}) \prod_{i \notin I} \Pr_{R_i} \right) [u^i \mapsto u^{i+1} - u^{i-1}]
\end{aligned}$$

hence

$$\begin{aligned}
&\sum_{u^{r+1},u^r} \left| \Pr_{F_1,\ldots,F_r}[u^0u^1 \mapsto u^{r+1}u^r] - \Pr_{R_1,\ldots,R_r}[u^0u^1 \mapsto u^{r+1}u^r] \right| \\
&\leq \sum_{u^2,\ldots,u^{r+1}} \sum_{\substack{I \subseteq \{1,\ldots,r\} \\ I \neq \emptyset}} \left(\prod_{i \in I} |\Pr_{F_i} - \Pr_{R_i}| \prod_{i \notin I} \Pr_{R_i} \right) [u^i \mapsto u^{i+1} - u^{i-1}] \\
&\leq \sum_{\substack{I \subseteq \{1,\ldots,r\} \\ I \neq \emptyset}} \prod_{i \in I} \epsilon_i \\
&= (1+\epsilon_1)\ldots(1+\epsilon_r) - 1.
\end{aligned}$$

□

From this lemma and the previous observation we obtain the following theorem.

Theorem F.12. *Let $F_1, \ldots, F_r$ be r independent random functions on $\mathcal{M}_0$ such that ${}^f\mathrm{Dec}^d_{|||.|||_\infty}(F_i) \leq \epsilon$ $(i = 1, \ldots, r)$. For any $k \geq 3$ we have*

$$\mathrm{Dec}^d_{|||.|||_\infty}(\Psi(F_1, \ldots, F_r)) \leq \left((1+\epsilon)^k - 1 + \frac{2d^2}{\sqrt{\#\mathcal{M}}} \right)^{\lfloor \frac{r}{k} \rfloor}.$$

We can remark that the lemma remains valid if we replace the group operation used in the Feistel construction by any other pseudogroup law. This makes the decorrelation $|||.|||_\infty$-bias a friendly tool for constructing Feistel Ciphers.

Input: a cipher c, a complexity n, a characteristic (a, b)

1. for i from 1 to n do
   (a) pick uniformly a random X and query for $c(X)$ and $c(X + a)$
   (b) if $c(X + a) = c(X) + b$, stop and output 1
2. output 0

FIGURE F.3 – Differential Distinguisher.

F.6 Differential Cryptanalysis

In this section we assume that $\mathcal{M}$ is given a group structure of order M. We study the security of pairwise decorrelated ciphers against basic differential cryptanalysis. We study criteria which prove that the attack cannot be better than exhaustive attack, M.

Let C be a cipher on $\mathcal{M}$ and let C^* be the Perfect Cipher.

Although differential cryptanalysis has been invented in order to recover a whole key by Biham and Shamir (see [29, 30]), we study here the basic underlying notion which makes it work. We call basic differential cryptanalysis the distinguisher which is characterized by a pair $(a, b) \in \mathcal{M}^2$ with $a \neq 0$ and which is depicted on Fig. F.3.

It is well known that differential cryptanalysis depends on the following $\mathrm{DP}^C(a, b)$ (see for instance [110]). We define

$$\mathrm{DP}^C(a, b) = \Pr_X[C(X + a) = C(X) + b]$$

where X has a uniform distribution. This quantity thus depends on the choice of the cipher (*i.e.* on the key). Here we focus on average complexities of attacks with no prior information on the key, *i.e.* on the average value $E\left(\mathrm{DP}^C(a, b)\right)$ over the distribution of C. The problem of successful attacks for sets of *weak keys* is not our purpose here. We first mention that $E\left(\mathrm{DP}^C(a, b)\right)$ has an interesting linear expression with respect to the pairwise distribution matrix of C. Namely, straightforward computation shows that

$$E(\mathrm{DP}^C(a, b)) = \frac{1}{M} \sum_{\substack{x_1, x_2 \\ y_1, y_2}} 1_{\substack{x_2 = x_1 + a \\ y_2 = y_1 + b}} \Pr \begin{bmatrix} x_1 & \overset{C}{\mapsto} & y_1 \\ x_2 & \mapsto & y_2 \end{bmatrix}. \tag{F.4}$$

Lemma F.13. *For the attack of Fig. F.3, we have*

$$\mathrm{Adv}_{\mathrm{Fig.F.3}}(C) \leq n. \max\left(\frac{1}{M-1}, E\left(\mathrm{DP}^C(a, b)\right)\right).$$

Proof. It is straightforward to see that the probability, for some fixed key, that the attack accepts C is

$$1 - \Big(1 - \mathrm{DP}^C(a,b)\Big)^n$$

which is less than $n.\mathrm{DP}^C(a,b)$. Hence we have $p \leq n.E\left(\mathrm{DP}^C(a,b)\right)$. Since from Equation (F.4) we have $E\left(\mathrm{DP}^{C^*}(a,b)\right) = \frac{1}{M-1}$, we obtain the result. □

Theorem F.14. *Let C be a cipher on a group $\mathcal{M}$ of order M. For any basic differential distinguisher (depicted on Fig. F.3) of complexity n, we have*

$$\mathrm{Adv}_{\mathrm{Fig.F.3}}(C) \leq \frac{n}{M-1} + \frac{n}{2}\mathrm{Dec}^2_{|||\cdot|||_\infty}(C).$$

Proof. Actually we notice that $E\left(\mathrm{DP}^{C^*}(a,b)\right) = \frac{1}{M-1}$ and that

$$\left|E\left(\mathrm{DP}^C(a,b)\right) - \frac{1}{M-1}\right| \leq \frac{1}{2}\mathrm{Dec}^2_{|||\cdot|||_\infty}(C)$$

from Equation (F.4). □

So, if the pairwise decorrelation bias has the order of $1/M$, basic differential cryptanalysis does not work against C, but with a complexity in the scale of M.

F.7 Linear Cryptanalysis

Linear cryptanalysis has been invented by Matsui [89, 91] based on the notion of statistical attacks which are due to Gilbert *et al.* [54, 158, 53]. We study here the simpler version of the original attack against pairwise decorrelated ciphers.

In this section we assume[2] that $\mathcal{M} = \mathrm{GF}(2^m)$. The inner dot product $a \cdot b$ in $\mathrm{GF}(2^m)$ is the parity of the bitwise *and* of a and b.

Let C be a cipher on $\mathcal{M}$ and let C^* be the Perfect Cipher.

As in Section F.6, we similarly call basic linear cryptanalysis the distinguisher characterized by a pair $(a,b) \in \mathcal{M}^2$ with $b \neq 0$ which is depicted on Fig. F.4.

We notice here that the attack depends on the way it accepts or rejects depending on the final counter c value.

[2]Although it is easy to generalize the notion of linear cryptanalysis over other finite fields, we only consider the characteristic 2 case for a better understanding.

Input: a cipher c, a complexity n, a characteristic (a, b), a set A

1. initialize the counter value u to zero
2. for i from 1 to n do
   (a) pick a random X with a uniform distribution and query for $c(X)$
   (b) if $X \cdot a = c(X) \cdot b$, increment the counter u
3. if $u \in A$, output 1, otherwise output 0

FIGURE F.4 – Linear Distinguisher.

As pointed out by Chabaud and Vaudenay [35], linear cryptanalysis is based on the quantity

$$\mathrm{LP}^C(a,b) = \left(2 \Pr_X[X \cdot a = C(X) \cdot b] - 1\right)^2.$$

(Here we use Matsui's notations taken from [92].) As for differential cryptanalysis, we focus on $E\left(\mathrm{LP}^C(a,b)\right)$, and there is a linear expression of this mean value in term of the pairwise distribution matrix $[C]^2$ which comes from straightforward computations :

$$E(\mathrm{LP}^C(a,b)) = 1 - 2^{2-2m} \sum_{\substack{x_1 \neq x_2 \\ y_1 \neq y_2}} 1_{\substack{x_1 \cdot a = y_1 \cdot b \\ x_2 \cdot a \neq y_2 \cdot b}} \Pr\begin{bmatrix} x_1 & \overset{C}{\mapsto} & y_1 \\ x_2 & \mapsto & y_2 \end{bmatrix}. \tag{F.5}$$

Lemma F.15. *For the attack of Fig. F.4 we have*

$$\lim_{n \to +\infty} \frac{\mathrm{Adv}_{\mathrm{Fig.F.4}}(C)}{n^{\frac{1}{3}}} \leq 9.3 \left(\max\left(\frac{1}{2^m - 1}, E\left(\mathrm{LP}^C(a,b)\right)\right)\right)^{\frac{1}{3}}.$$

Proof. Let N_i be the random variable defined as being 1 or 0 depending on whether or not we have $x \cdot a = c(x) \cdot b$ in the ith iteration. All N_i's are independent and with the same 0-or-1 distribution. Let μ be the probability that $N_i = 1$, for a fixed permutation c. From the Central Limit Theorem, we can approximate the final quantity u/n to a normal distribution law with mean μ and standard deviation $\sigma = \sqrt{\frac{\mu(1-\mu)}{n}}$. Let A be the set of all accepted u/n quantities. For a fixed c, the probability that the attack accepts is

$$p^c \underset{n \to +\infty}{\sim} \int_{t \in A} \frac{e^{-\frac{(t-\mu)^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}} dt.$$

We let p_{eq}^c denotes the right hand term of this equation. We can compare it to the theoretical expected value p_0 of p_{eq}^c defined by $\mu = \frac{1}{2}$ and $\sigma = \frac{1}{2\sqrt{n}}$ *i.e.*

$$p_0 = \int_{t \in A} \frac{e^{-\frac{(t-\frac{1}{2})^2}{n}}}{\sqrt{2\pi}} 2\sqrt{n} dt.$$

The difference $p_{\text{eq}}^c - p_0$ is maximal when $A = [\tau_1, \tau_2]$ for some values τ_1 and τ_2 which are roots of the Equation

$$\frac{(t-\mu)^2}{\sigma^2} + \log \sigma^2 = 4n \left(t - \frac{1}{2}\right)^2 - \log 4n.$$

Hence, the maximum of the difference $p_{\text{eq}}^c - p_0$ is at most the maximum when $A = [\tau_1, \tau_2]$ over the choice of τ_1 and τ_2, which is the maximum minus the minimum of $p_{\text{eq}}^c - p_0$ when $A =]-\infty, \tau]$. Now we have

$$\int_{-\infty}^{\tau} \frac{e^{-\frac{(t-\mu)^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}} dt = \int_{-\infty}^{\frac{\tau-\mu}{\sigma}} \frac{e^{-\frac{t^2}{2}}}{\sqrt{2\pi}} dt$$

so we have

$$p_{\text{eq}}^c - p_0 \leq \left(\max_{\tau} - \min_{\tau}\right) \int_{2\sqrt{n}(\tau-\frac{1}{2})}^{\sqrt{n}\frac{\tau-\mu}{\sqrt{\mu(1-\mu)}}} \frac{e^{-\frac{t^2}{2}}}{\sqrt{2\pi}} dt.$$

We consider the sum as a function $f(\mu)$ on μ. Since we have $f\left(\frac{1}{2}\right) = 0$, we have $|f(\mu)| \leq B.\left|\mu - \frac{1}{2}\right|$ where B is the maximum of $|f'(x)|$ when x varies from μ to $\frac{1}{2}$. We have

$$f'(x)\sqrt{\frac{2\pi}{n}} = \left(-\frac{1}{\sqrt{x(1-x)}} - \frac{\frac{1}{2}-x}{x(1-x)} \frac{\tau - x}{\sqrt{x(1-x)}}\right) e^{-\frac{n(\tau-x)^2}{2x(1-x)}}$$

so

$$|f'(x)| \leq \sqrt{\frac{n}{2\pi\mu(1-\mu)}} + \frac{\left|\mu - \frac{1}{2}\right|}{\mu(1-\mu)} \frac{e^{-\frac{1}{2}}}{\sqrt{2\pi}} \leq \sqrt{\frac{n}{2\pi\mu(1-\mu)}} + \frac{\left|\mu - \frac{1}{2}\right|}{\mu(1-\mu)}.$$

Therefore we have

$$\left|p_{\text{eq}}^c - p_0\right| \leq 2\sqrt{\frac{n}{2\pi}} \frac{\left|\mu - \frac{1}{2}\right|}{\sqrt{\mu(1-\mu)}} + 2\frac{\left(\mu - \frac{1}{2}\right)^2}{\mu(1-\mu)}. \tag{F.6}$$

Let $\delta = E\left((2\mu - 1)^2\right)$ over the distribution of C. (We recall that μ depends on the permutation c.) Let $\alpha = \frac{1}{8}\left(\delta\sqrt{\frac{2\pi}{n}}\right)^{\frac{1}{3}}$. Since $\delta \leq 1$ and $n \geq 1$, we have $\alpha \leq .17$ so if $\left|\mu - \frac{1}{2}\right| \leq \alpha$ we have

$$\left|p_{\text{eq}}^c - p_0\right| \leq .55(\delta n)^{\frac{1}{3}}.$$

Now we have $\left|\mu - \frac{1}{2}\right| \geq \alpha$ with a probability less than $\frac{\delta}{4\alpha^2}$, which is less than $8.68(\delta n)^{\frac{1}{3}}$, and in this case we have $\left|p_{\text{eq}}^c - p_0\right| \leq 1$. Hence, we have

$$E\left(\left|p_{\text{eq}}^c - p_0\right|\right) \leq 9.3(\delta n)^{\frac{1}{3}}.$$

We note that $\delta = E\left(\text{LP}^C(a,b)\right)$. We finally note that $E\left(\text{LP}^{C^*}(a,b)\right) = \frac{1}{2^m-1}$ from Equation (F.5). □

Theorem F.16. *Let C be a cipher on $\mathcal{M} = \{0,1\}^m$. For any linear distinguisher (depicted on Fig. F.4) we have*

$$\lim_{n\to+\infty} \frac{\text{Adv}_{\text{Fig.F.4}}(C)}{n^{\frac{1}{3}}} \leq 9.3\left(\frac{1}{2^m-1} + 2\text{Dec}^2_{|||\cdot|||_\infty}(C)\right)^{\frac{1}{3}}.$$

This asymptotic result comes from approximation to the normal law by the Large Number Theorem, which is correct whenever n is not too small (*i.e.* $n > 30$). This result is thus actually valid for any practical n.

Proof. Actually we notice that $E\left(\text{LP}^{C^*}(a,b)\right) = \frac{1}{2^m-1}$ and that

$$\left|E\left(\text{LP}^C(a,b)\right) - \frac{1}{2^m-1}\right| \leq 2\text{Dec}^2_{|||\cdot|||_\infty}(C)$$

from Equation (F.5). □

So, if the pairwise decorrelation bias has the order of 2^{-m}, linear distinguishers does not work against C, but with a complexity in the scale of 2^m.

F.8 Iterated Attacks of Order d

Theorems F.14 and F.16 suggest that we try to generalize them to attacks in the model depicted on Fig. F.5. In this model, we iterate a d-limited non-adaptive attack $\mathcal{T}$. We assume that this attack obtains a sample (X, Y) with $X = (X_1, \ldots, X_d)$ and $Y = (Y_1, \ldots, Y_d)$ such that $y_i = c(X_i)$ for a given distribution of X. Thus, we can think of a known plaintext attack where X

Input: a cipher c, a complexity n, a distribution on X, a test $\mathcal{T}$, an acceptance set A

1. for i from 1 to n do
   (a) get a new $X = (X_1, \ldots, X_d)$
   (b) get $Y = (c(X_1), \ldots, c(X_d))$
   (c) set $T_i = 0$ or 1 with an expected value $\mathcal{T}(X, Y)$
2. if $(T_1, \ldots, T_n) \in A$ output 1 otherwise output 0

Figure F.5 – Iterated Attack of Order d.

has a fixed distribution (*e.g.* a uniform distribution) or of a chosen plaintext attack where X has a given distribution (*e.g.* in differential cryptanalysis, $X = (X_1, X_1 + a)$ where X_1 has a uniform distribution). The result of the attack depends on the result of all iterated ones in a way characterized by a set A. For instance, if $A = \{0,1\}^n \backslash \{(0, \ldots, 0)\}$ we can define the differential cryptanalysis (thus of order $d = 2$). If A is the set of all $(t_1, \ldots, t_n)$ with an acceptable sum we can define the linear cryptanalysis (of order $d = 1$).

It is tenting to believe that a cipher resists to this model of attacks once it has a small d-wise decorrelation bias. This is wrong as the following example shows. Let C be a cipher with a perfect d-wise decorrelation. We assume that an instance c of C is totally defined by d (x_i, y_i) points so that C is uniformly distributed in a set of $K = M(M-1)\ldots(M-d+1)$ permutations denoted $c_1, \ldots, c_K$. From $x = (x_1, \ldots, x_d)$ and $y = (y_1, \ldots, y_d)$ we can define $I(x, y)$ as the unique index k such that $c_k(x_i) = y_i$ for $i = 1, \ldots, d$. We let

$$\mathcal{T}(x, y) = \begin{cases} 1 & \text{if } I(x, y) \equiv 0 \bmod \mu \\ 0 & \text{otherwise} \end{cases}$$

for a given modulus $\mu = n/a$ and

$$\mathcal{A} = \{0,1\}^n \backslash \{(0, \ldots, 0)\}.$$

If we feed this attack with C or C^*, we have

$$p \approx \frac{1}{\mu} = \frac{a}{n} \quad \text{and} \quad p^* \approx 1 - \left(1 - \frac{1}{\mu}\right)^n \approx 1 - e^{-a}$$

for $a << n$. Thus Adv can be large even with a relatively large n. This problem actually comes from the fact that the tests $\mathcal{T}$ provide a same expected result for C and C^* but a totally different standard deviation.

We can however prove the security when the cipher has a good decorrelation to the order $2d$.

Theorem F.17. *Let C be a cipher on a message space of size M such that $\mathrm{Dec}^{2d}_{|||.|||_\infty}(C) \leq \epsilon$ for some given $d \leq M/2$. For any iterated attack (depicted on Fig. F.5) of order d such that the obtained plaintexts are independent, we have*

$$\mathrm{Adv}_{\mathrm{Fig.F.5}}(C) \leq 3\left(\left(2\delta + \frac{5d^2}{2M} + \frac{3\epsilon}{2}\right)n^2\right)^{\frac{1}{3}} + \frac{n\epsilon}{2}$$

where δ is the probability that for two independent X and X' there exists i and j such that $X_i = X'_j$.

For instance, if the distribution of X is uniform, we have $\delta \leq \frac{d^2}{2M}$.

Proof. Let Z (resp. Z^*) be the probability that the test accepts $(X, C(X))$ (resp. $(X, C^*(X))$), *i.e.*

$$Z = E_X(\mathcal{T}(X, C(X))).$$

Let p (resp. p^*) be the probability that the attack accepts, *i.e.*

$$p = \Pr_C[(T_1, \ldots, T_n) \in A].$$

Since the T_is are independent and with the same expected value Z which only depends on C, we have

$$p = E_C\left(\sum_{(t_1,\ldots,t_n)\in A} Z^{t_1+\ldots+t_n}(1-Z)^{n-(t_1+\ldots+t_n)}\right).$$

We thus have $p = E(f(Z))$ where $f(z)$ is a polynomial of degree at most n with values in $[0,1]$ for any $z \in [0,1]$ entries and with the form $f(z) = \sum a_i z^{b_i}(1-z)^{n-b_i}$. It is straightforward that $|f'(z)| \leq n$ for any $z \in [0,1]$. Thus we have $|f(z) - f(z^*)| \leq n|z - z^*|$.

The crucial point in the proof is in proving that $|Z - Z^*|$ is small within a high probability. For this, we need $|E(Z) - E(Z^*)|$ and $|V(Z) - V(Z^*)|$ to be both small.

From Theorem F.7 we know that $|E(Z) - E(Z^*)| \leq \frac{\epsilon}{2}$. We note that Z^2 corresponds to a another test but with $2d$ entries, hence we have $|E(Z^2) - E((Z^*)^2)| \leq \frac{\epsilon}{2}$. Hence $|V(Z) - V(Z^*)| \leq \frac{3}{2}\epsilon$. Now from the Tchebichev's Inequality we have

$$\Pr[|Z - E(Z)| > \lambda] \leq \frac{V(Z)}{\lambda^2}.$$

Hence we have

$$|p - p^*| \leq \frac{2V(Z^*) + \frac{3}{2}\epsilon}{\lambda^2} + n\left(\frac{\epsilon}{2} + 2\lambda\right)$$

so, with $\lambda = \left(\frac{2V(Z^*)+\frac{3}{2}\epsilon}{n}\right)^{\frac{1}{3}}$ we have

$$|p - p^*| \leq 3\left(\left(2V(Z^*) + \frac{3\epsilon}{2}\right)n^2\right)^{\frac{1}{3}} + \frac{n\epsilon}{2}.$$

Now we have

$$\begin{aligned} V(Z^*) &= \sum_{\substack{(x,y)\in A \\ (x',y')\in A}} \Pr_X[x]\Pr_X[x']\left(\Pr_{C^*}\left[\begin{array}{ccc} x & \to & y \\ x' & \to & y' \end{array}\right] - \Pr_{C^*}[x\to y]\Pr_{C^*}[x'\to y']\right) \\ &\leq \frac{1}{2}\sum_{\substack{x,y \\ x',y'}} \Pr_X[x]\Pr_X[x']\left|\Pr_{C^*}\left[\begin{array}{ccc} x & \to & y \\ x' & \to & y' \end{array}\right] - \Pr_{C^*}[x\to y]\Pr_{C^*}[x'\to y']\right|. \end{aligned}$$

The sum over all x and x' entries with colliding entries (*i.e.* with some $x_i = x'_j$) is less than δ. The sum over all y and y' entries with colliding entries and no colliding x and x' is less than $d^2/4M$. The sum over all no colliding x and x' and no colliding y and y' is equal to

$$\frac{1-\delta}{2}\left(1 - \frac{M(M-1)\ldots(M-2d+1)}{M^2(M-1)^2\ldots(M-d+1)^2}\right)$$

which is less than $\frac{d^2}{2(M-d)}$. Thus we have $V(Z^*) \leq \delta + \frac{d^2}{4M} + \frac{d^2}{2(M-d)}$ which is less than $\delta + \frac{5d^2}{4M}$ when $2d \leq M$.. □

This theorem proves that we need $n = \Omega(1/\sqrt{\epsilon})$ or $n = \Omega(\sqrt{M})$ to have a meaningful iterated attack. If we apply it to linear cryptanalysis, this result is thus weaker than Theorem F.16. It is however much more general.

F.9 COCONUT: a Perfect Decorrelation Design

In this section we define the COCONUT Ciphers family which are perfectly decorrelated ciphers to the order two.

The COCONUT Ciphers are characterized by some parameters (m, p). m is the block length, and p is a irreducible polynomial of degree m in GF(2) (which defines a representation of the GF(2^m) Galois Field). A COCONUT Cipher of block length m is simply a product cipher $C_1 \circ C_2 \circ C_3$ where C_1 and C_3 are any (possibly weak) ciphers which can depend from each other, and C_2 is an independent cipher based on a $2m$-bit key which consists of two

polynomials A and B of degree at most $m-1$ over GF(2) such that $A \neq 0$. For a given representation of polynomials into m-bit strings, we simply define

$$C_2(x) = A.x + B \bmod p.$$

Since C_2 performs perfect decorrelation to the order two and since it is independent from C_1 and C_3, any COCONUT Cipher is obviously perfectly decorrelated to the order two. Therefore Theorems F.14 and F.16 shows that COCONUT resists to the basic differential and linear cryptanalysis.

One can wonder why C_1 and C_3 are for. Actually, C_2 makes some precise attacks provably impractical, but in a way which makes the cipher obviously weak against other attacks. (C_2 is actually a linear function, thus although we can prove it resists to some attacks which are characterized by some parameter $d \leq 2$, it is fairly weak against attacks of $d = 3$.) We believe that all real attacks on any real cipher have an intrinsic *order* d, that is they use the d-wise correlation in the encryption of d messages. Attacks of a large d on real ciphers are impractical, because the d-wise decorrelation can hardly be analyzed since it depends on too many factors. Therefore, the COCONUT approach consists in making the cipher provably resistant against attacks of order at most 2 such as differential or linear cryptanalysis, and heuristically secure against attacks of higher order by real life ciphers as C_1 and C_3.

The COCONUT98 Cipher has been proposed in [173] with parameters $m = 64$ and $p = x^{64} + x^{11} + x^2 + x + 1$.

F.10 PEANUT: a Partial Decorrelation Design

In this section we define the PEANUT Ciphers family, which achieves an example of partial decorrelation. This family is based on a combinatorial tool which has been previously used by Halevi and Krawczyk for authentication in [57].

The PEANUT Ciphers are characterized by some parameters (m, r, d, p). They are Feistel Ciphers of block length of m bits (m even), r rounds. The parameter d is the order of partial decorrelation that the cipher performs, and p must be a prime number greater than $2^{\frac{m}{2}}$.

The cipher is defined by a key of $\frac{mrd}{2}$ bits which consists of a sequence of r lists of d $\frac{m}{2}$-bit numbers, one for each round. In each round, the F function has the form

$$F(x) = g(k_1.x^{d-1} + k_2.x^{d-2} + \ldots + k_{d-1}.x + k_d \bmod p \bmod 2^{\frac{m}{2}})$$

where g is any permutation on the set of all $\frac{m}{2}$-bit numbers.

Let us now estimate the decorrelation $|||.|||_\infty$-bias of the PEANUT ciphers.

Lemma F.18. *Let $\mathbf{K} = \mathrm{GF}(q)$ be a finite field, let $r : \{0,1\}^{\frac{m}{2}} \to \mathbf{K}$ be an injective mapping, and let $\pi : \mathbf{K} \to \{0,1\}^{\frac{m}{2}}$ be a surjective mapping. Let F be a random function defined by*

$$F(x) = \pi(r(A_{d-1}).r(x)^{d-1} + \ldots + r(A_0))$$

where the A_i's are independent and uniformly distributed in $\{0,1\}^{\frac{md}{2}}$. We have

$${}^f\mathrm{Dec}^d_{|||.|||_\infty}(F) \leq 2\left(\left(\frac{q}{2^{\frac{m}{2}}}\right)^d - 1\right).$$

Proof. Let $x = (x_1, \ldots, x_d)$ be a multi-point in $\{0,1\}^{\frac{m}{2}}$. We want to prove that

$$S = \sum_{y=(y_1,\ldots,y_d)} |[F]^d_{x,y} - [F^*]^d_{x,y}| \leq 2\left(\left(\frac{q}{2^{\frac{m}{2}}}\right)^d - 1\right).$$

Let c be the number of pairwise different x_i's. For any y such that there exists (i,j) such that $y_i \neq y_j$ and $x_i = x_j$, the contribution to the sum is zero. So we can assume that y is defined over the $2^{\frac{cm}{2}}$ choices of y_i's on positions corresponding to pairwise different x_i's. If we let $x_{d+1}, \ldots, x_{2d-c}$ be new fixed points such that we have exactly d pairwise different x_i's, since the probability that F (resp. F^*) maps x onto y is equal to the sum over all choices of $y_{d+1}, \ldots, y_{2d-c}$ that it maps the extended x onto the extended y, we can assume w.l.o.g. that $c = d$.

For any multi-point y we thus have that $\Pr[x \mapsto y] = j.2^{-\frac{md}{2}}$ where j is an integer. Let N_j be the number of multi-points y which verify this property. We have $\sum_j N_j = 2^{\frac{md}{2}}$ and $\sum_j jN_j = 2^{\frac{md}{2}}$. We have

$$S \leq \sum_j N_j \left|\frac{j-1}{2^{\frac{md}{2}}}\right| = 2N_0.2^{-\frac{md}{2}}.$$

N_0 is the number of unreachable y's, *i.e.* the y's which correspond to a polynomial whose coefficients are not all r-images. This number is thus less than the number of missing polynomials which is $q^d - 2^{\frac{md}{2}}$. □

From Theorem F.12 with $k = 3$ we thus obtain the following theorem.

Theorem F.19. *Let C be a cipher in the PEANUT family with parameters (m, r, d, p). We have*

$$\mathrm{Dec}^d_{|||\cdot|||_\infty}(C) \leq \left(\left(1 + 2\left(p^d 2^{-\frac{md}{2}} - 1\right)\right)^3 - 1 + \frac{2d^2}{2^{\frac{m}{2}}}\right)^{\lfloor \frac{r}{3} \rfloor}.$$

When $p \approx 2^{\frac{m}{2}}$ we can approximate

$$\mathrm{Dec}^{d}_{|||.|||_\infty}(C) \approx \left(\frac{6d\left(p - 2^{\frac{m}{2}}\right) + 2d^2}{2^{\frac{m}{2}}} \right)^{\lfloor \frac{r}{3} \rfloor}.$$

Example F.20. We can use the parameters $m = 64$, $r = 9$, $d = 2$ and $p = 2^{32} + 15$. We obtain that $\mathrm{Dec}^{2}_{|||.|||_\infty}(C) \leq 2^{-76}$. Therefore from Theorems F.14 and F.16 no differential or linear distinguisher can be efficient. The PEANUT98 Cipher has been proposed with these parameters in [173].

In an earlier version of this work [171], we proposed a similar construction (say PEANUT97) which uses prime numbers smaller than $2^{\frac{m}{2}}$. However the result above does not hold with the $|||.|||_\infty$ norm, but rather with the $||.||_2$ one. The drawback is that this norm has less friendly theorems for constructing Feistel ciphers, and in particular we need more rounds to make the cipher provably secure. (For more information, see [176].)

F.11 Conclusion and Further Work

Decorrelation modules are cheap and friendly tools which can strengthen the security of block ciphers. Actually, we can quantify their security against a class of cryptanalysis which includes differential and linear cryptanalysis. To illustrate this paradigm, we proposed two definite prototype ciphers in [173].

Research on other general cryptanalysis is still an open problem. In particular, it is not sure that $2d$-decorrelation is necessary for getting provable security against iterated attacks of order d (Theorem F.17).

One problem with the COCONUT or PEANUT construction is that it requires a long key (in order to make the internal random functions independent).

generator fed with a short key, but the results on the security based on decorrelation are no longer valid. However, provided that the pseudorandom generator produces outputs which are indistinguishable from truly random sequences, we can still prove the security. This approach has been developed in [55, 56] for submitting a candidate (DFC) to the *Advanced Encryption Standard* process which has been initiated by the U.S. Government.

Acknowledgments

I wish to thank Jacques Stern for valuable help. I also thank the CNRS for having honored this work as one of the 100 “important facts” in engineering advances in France (see [10]).

Annexe G

CS-Cipher

[*Cet article de* Jacques Stern *et* Serge Vaudenay *a été publié dans* Fast Software Encryption *98 [155].*]

Abstract. This paper presents a new block cipher which offers good encryption rate on any platform. It is particularly optimized for hardware implementation where the expected rate is several Gbps on a small dedicated chip working at 30MHz. Its design combines up to date state of the art concepts in order to make it (hopefully) secure: diffusion network based on the Fast Fourier Transform, multipermutations, highly nonlinear confusion boxes.

Recent explosion of the telecommunication marketplace motivates the research on encryption schemes. Trading security issues pushed the US government to start the development of the *Data Encryption Standard* in the 70's [2], all telecommunication devices now need to be secured by encryption. Many attacks have been proposed against DES including Biham and Shamir's differential cryptanalysis [29, 30] and Matsui's linear cryptanalysis [89, 91]. Still the best practical attack seems to be exhaustive search, which has become a real threat as shown by the recent success of the RSA Challenge [182]. In this paper we propose a new symmetric encryption scheme which has been designed in order to be efficient on any platform, included cheap 8-bit microprocessors (*e.g.* smart cards), modern 32-bit microprocessors (SPARC, Pentium) and dedicated chips.

Notations

- $||$ is the concatenation of two strings

- $\oplus$ is the bitwise exclusive *or* of two bitstrings (with equal lengths)
- R_l rotates a bitstring by one position to the left
- $\wedge$ is the bitwise *and* of two bitstrings (with equal lengths)
- bitstrings are written in hexadecimal by packing four bits into one digit (for instance, $\texttt{d2}_x$ denotes the bitstring `11010010`)
- the numbering of bits in bitstrings is from right to left starting with 0 (*i.e.* x_0 denotes the last bit in x)
- bitstring and integers are converted in such a way that $b_{n-1}||\ldots||b_0$ corresponds to an integer $b_{n-1}.2^{n-1} + \ldots + b_0$

G.1 Definition of CS-CIPHER

G.1.1 Use of CS-CIPHER

CS-CIPHER (as for the French "*Chiffrement Symétrique*", *Symmetric Cipher*) is a symmetric block cipher which can be used in any mode to encrypt a block stream (*e.g.* the Cipher Block Chaining mode, see [3]). Basically, the CS-CIPHER encryption function maps a 64-bit plaintext block m onto a 64-bit ciphertext block m' by using a secret key k which is a bitstring with arbitrary length up to 128. The CS-CIPHER decryption function maps the ciphertext onto the plaintext by using the same secret key. We assume that m is represented by a bitstring

$$m = m_{63} \ldots m_1 m_0$$

and we similarly write

$$m' = m'_{63} \ldots m'_1 m'_0.$$

We also assume that the string k is padded with trailing zero bits to get a length of 128 bits

$$k = k_{127} \ldots k_1 k_0.$$

(A key k is therefore equivalent to another key k' which consists in padding k with a few zero bits.)

A key scheduling scheme first process the secret key k in order to obtain nine 64-bit subkeys $k^0, \ldots, k^8$ iteratively in this order. If the secret key has to be used several times, we recommend to precompute this sequence which may notably increase the encryption rate.

The encryption algorithm processes iteratively each subkey in the right order $k^0, \ldots, k^8$ whereas the decryption algorithm processes them in the reverse order $k^8, \ldots, k^0$. We thus recommend to keep a storage of all subkeys for decryption or to adapt the key scheduling scheme so that it can generate the subkeys in the reverse order.

G.1.2 Key Scheduling Scheme

Let k be the padded 128-bit secret key. We first split the bitstring into two 64-bit strings denoted k^{-2} and k^{-1} such that

$$k = k^{-1}||k^{-2}.$$

Those strings initialize a sequence $k^{-2}, \ldots, k^8$ where $k^0, \ldots, k^8$ are the nine 64-bit subkeys to compute. The sequence comes from a Feistel scheme as

$$k^i = k^{i-2} \oplus F_{c^i}(k^{i-1})$$

for $i = 0, \ldots, 8$ where F_{c^i} is defined below (see Feistel [51]). Figure G.1 illustrates the key scheduling scheme together with the encryption itself.

The F_{c^i} function maps a 64-bit string onto a 64-bit string by using a 64-bit constant c^i. In the definition of CS-Cipher, $c^0, \ldots, c^8$ are defined as the first bytes of the table of a permutation P which will be defined below:

$$\begin{aligned}
c^0 &= \texttt{290d61409ceb9e8f}_x \\
c^1 &= \texttt{1f855f585b013986}_x \\
c^2 &= \texttt{972ed7d635ae1716}_x \\
c^3 &= \texttt{21b6694ea5728708}_x \\
c^4 &= \texttt{3c18e6e7faadb889}_x \\
c^5 &= \texttt{b700f76f73841163}_x \\
c^6 &= \texttt{3f967f6ebf149dac}_x \\
c^7 &= \texttt{a40e7ef6204a6230}_x \\
c^8 &= \texttt{03c54b5a46a34465}_x.
\end{aligned}$$

F_{c^i} is defined by

$$F_{c^i}(x) = T(P^8(x \oplus c^i)).$$

P^8 is defined by a byte-permutation P which maps an 8-bit string onto an 8-bit string according to a table and T is a bit transposition. (Sofware implementation will use a lookup table for P whereas hardware implementation may use the inner structure of P which will be detailed below.)

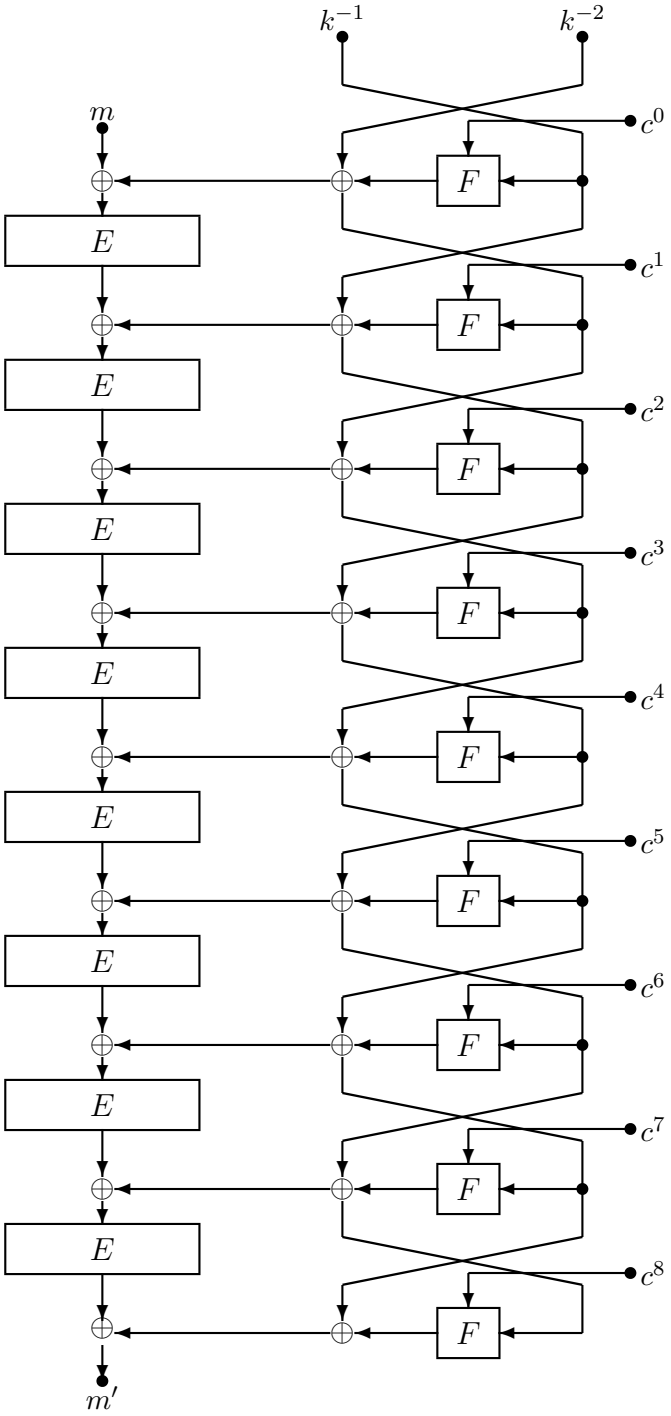


FIGURE G.1 – Encryption process

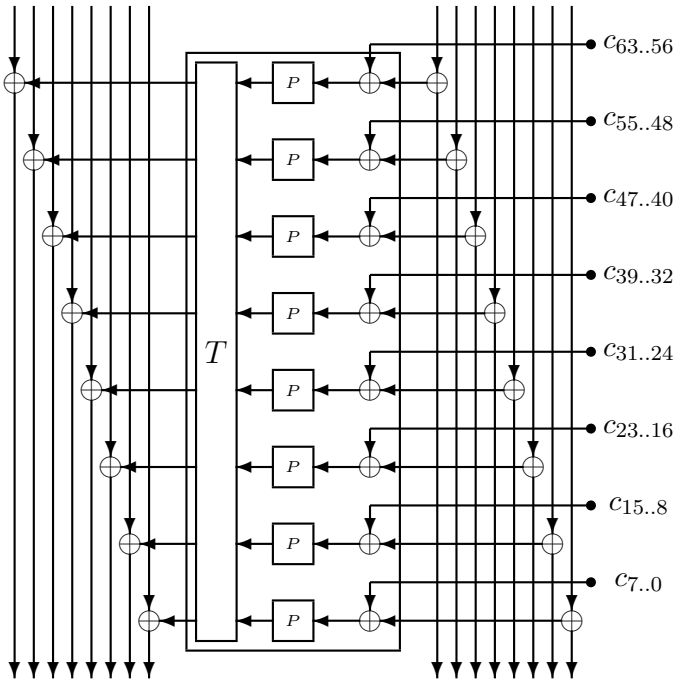


Figure G.2 – The F_{c^i} function in the key scheduling scheme

Given the 64-bit string $y = x \oplus c^i$, we split it into eight 8-bit strings denoted $y_{63..56}, \ldots, y_{7..0}$ such that $y = y_{63..56}|| \ldots ||y_{7..0}$. We next apply the permutation P byte-wise *i.e.* we compute

$$P^8(y_{63..56}|| \ldots ||y_{7..0}) = P(y_{63..56})|| \ldots ||P(y_{7..0}).$$

The permutation T is the 8×8 bit-matrix transposition. More precisely, given the 64-bit string $z = P^8(x \oplus c^i)$, we first split it into eight 8-bit strings $z_{63..56}, \ldots, z_{7..0}$ as for y above and write it in a 8×8 bit-matrix fashion in such a way that the first row is $z_{63..56}$ and so on. The permutation T simply transposes the matrix so that the first eight bits of $T(z)$ are the first bits of $z_{63..56}, \ldots, z_{7..0}$ in this order, the second eight bits are the seconds bits, and so on. Thus we have

$$T(z) = z_{63}||z_{55}|| \ldots ||z_7||z_{62}||z_{54}|| \ldots ||z_0.$$

Figure G.2 illustrates how F_{c^i} works in the key scheduling scheme.

G.1.3 Encryption Scheme

The encryption process is performed through eight rounds by using a round-encryption function E which is a permutation on the set of all 64-bit strings.

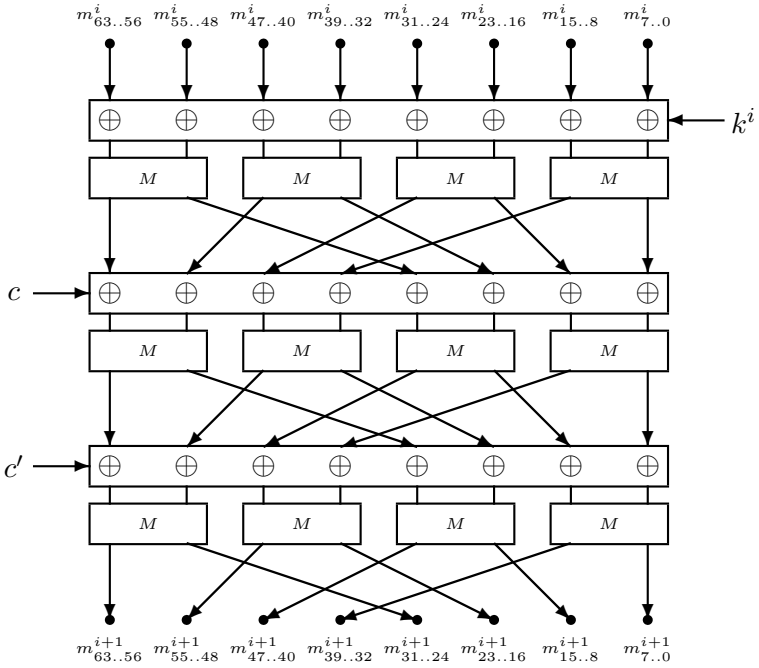


FIGURE G.3 – One encryption round

If m denotes the 64-bit plaintext block and $k^0, \ldots, k^8$ is the sequence of the 64-bit subkeys, the ciphertext block is

$$k^8 \oplus E(k^7 \oplus \ldots E(k^1 \oplus E(k^0 \oplus m)) \ldots)$$

as depicted on Figure G.1.

The round-encryption function E is based on the Fast Fourier Transform graph and a 16-bit to 16-bit *mixing* function M as depicted on Figure G.3. It also uses two 64-bit constants c and c' defined by the binary expansion of the mathematical constant

$$e = \sum_{n=0}^{\infty} \frac{1}{n!} = 2, \texttt{b7e151628aed2a6abf7158809cf4f3c762e7160f}_x \ldots$$

Thus we define

$$\begin{aligned} c &= \texttt{b7e151628aed2a6a} \\ c' &= \texttt{bf7158809cf4f3c7}. \end{aligned}$$

More precisely, in each encryption round, we iterate the following scheme three times

- we xor with a constant (which is successively the subkey k^i, c and c'),
- we split the 64-bit string into four 16-bit strings and we apply M to each of it, obtaining four 16-bit strings which combine into a 64-bit string,
- we split it again into eight 8-bit strings

  $$r_{63..56}||r_{55..48}||r_{47..40}||r_{39..32}||r_{31..24}||r_{23..16}||r_{15..8}||r_{7..0}$$

  and we change their order as

  $$r_{63..56}||r_{47..40}||r_{31..24}||r_{15..8}||r_{55..48}||r_{39..32}||r_{23..16}||r_{7..0}.$$

The M function takes a 16-bit string x which is split into two 8-bit strings $x_l||x_r$ and computes $M(x) = y_l||y_r$ by

$$\begin{aligned} y_l &= P(\varphi(x_l) \oplus x_r) \\ y_r &= P(R_l(x_l) \oplus x_r) \end{aligned}$$

where φ is defined by

$$\varphi(x_l) = (R_l(x_l) \wedge \mathtt{55}_x) \oplus x_l$$

i.e.

$$\varphi(x_7||\ldots||x_0) = x_7||(x_6 \oplus x_5)||x_5||(x_4 \oplus x_3)||x_3||(x_2 \oplus x_1)||x_1||(x_0 \oplus x_7).$$

The M computation is depicted on Figure G.4 (with the xor to its input which is always performed).

The P byte-permutation (which is also used in the key scheduling scheme) is defined by a three-round Feistel cipher represented on Figure G.5: the 8-bit input x is split into two 4-bit strings $x_l||x_r$, we compute successively

$$\begin{aligned} y &= x_l \oplus f(x_r) \\ z_r &= x_r \oplus g(y) \\ z_l &= y \oplus f(z_r) \end{aligned}$$

where f and g are two special functions.

The function f is defined by the table

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$f(x)$	f	d	b	b	7	5	7	7	e	d	a	b	e	d	e	f

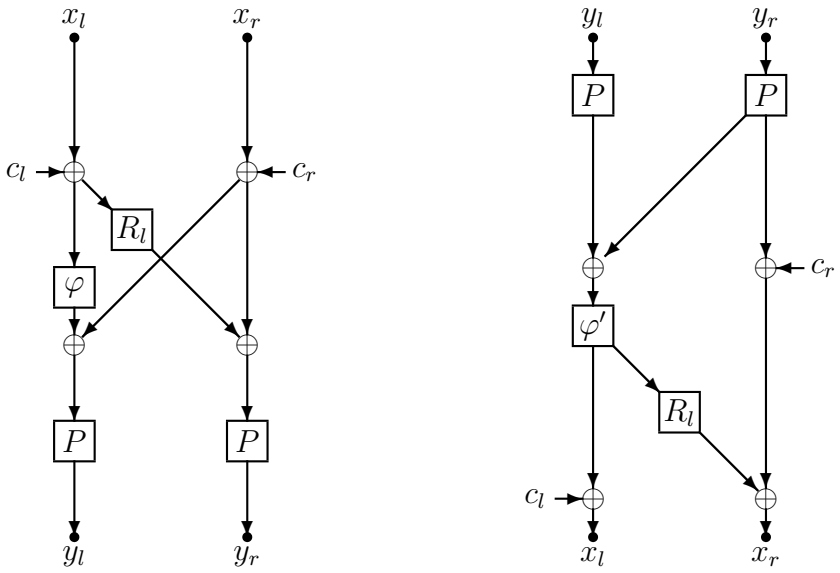


FIGURE G.4 – Computation graph of M and M^{-1}

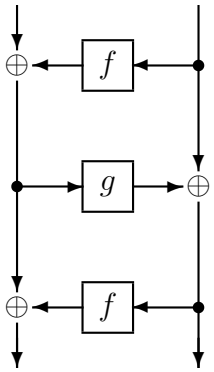


FIGURE G.5 – The permutation P

which comes from

$$f(x) = \overline{\overline{x} \wedge R_l(x)}.$$

The function g is defined by the table

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$g(x)$	a	6	0	2	b	e	1	8	d	4	5	3	f	c	7	9

which does not come from a simple expression.

Finally, the value of $P(xy)$ is given as follows by the table of P.

xy	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.a	.b	.c	.d	.e	.f
0.	29	0d	61	40	9c	eb	9e	8f	1f	85	5f	58	5b	01	39	86
1.	97	2e	d7	d6	35	ae	17	16	21	b6	69	4e	a5	72	87	08
2.	3c	18	e6	e7	fa	ad	b8	89	b7	00	f7	6f	73	84	11	63
3.	3f	96	7f	6e	bf	14	9d	ac	a4	0e	7e	f6	20	4a	62	30
4.	03	c5	4b	5a	46	a3	44	65	7d	4d	3d	42	79	49	1b	5c
5.	f5	6c	b5	94	54	ff	56	57	0b	f4	43	0c	4f	70	6d	0a
6.	e4	02	3e	2f	a2	47	e0	c1	d5	1a	95	a7	51	5e	33	2b
7.	5d	d4	1d	2c	ee	75	ec	dd	7c	4c	a6	b4	78	48	3a	32
8.	98	af	c0	e1	2d	09	0f	1e	b9	27	8a	e9	bd	e3	9f	07
9.	b1	ea	92	93	53	6a	31	10	80	f2	d8	9b	04	36	06	8e
a.	be	a9	64	45	38	1c	7a	6b	f3	a1	f0	cd	37	25	15	81
b.	fb	90	e8	d9	7b	52	19	28	26	88	fc	d1	e2	8c	a0	34
c.	82	67	da	cb	c7	41	e5	c4	c8	ef	db	c3	cc	ab	ce	ed
d.	d0	bb	d3	d2	71	68	13	12	9a	b3	c2	ca	de	77	dc	df
e.	66	83	bc	8d	60	c6	22	23	b2	8b	91	05	76	cf	74	c9
f.	aa	f1	99	a8	59	50	3b	2a	fe	f9	24	b0	ba	fd	f8	55

For instance, we have $P(26) = \mathsf{b8}$ since $f(6) = 7$, $2 \oplus 7 = 5$, $g(5) = \mathsf{e}$, $6 \oplus \mathsf{e} = 8$, $f(8) = \mathsf{e}$ and finally $5 \oplus \mathsf{e} = \mathsf{b}$.

G.1.4 Decryption Scheme

Decryption is performed by iterating a decryption-round function represented on Figure G.6.

Details of the decryption are left to the reader. We simply observe that $(x_l||x_r) = M^{-1}(y_l||y_r)$ can be computed by

$$\begin{aligned} x_l &= \varphi'(P(y_l) \oplus P(y_r)) \\ x_r &= R_l(x_l) \oplus P(y_r) \end{aligned}$$

where

$$\varphi'(x) = (R_l(x) \wedge \mathsf{aa}_x) \oplus x.$$

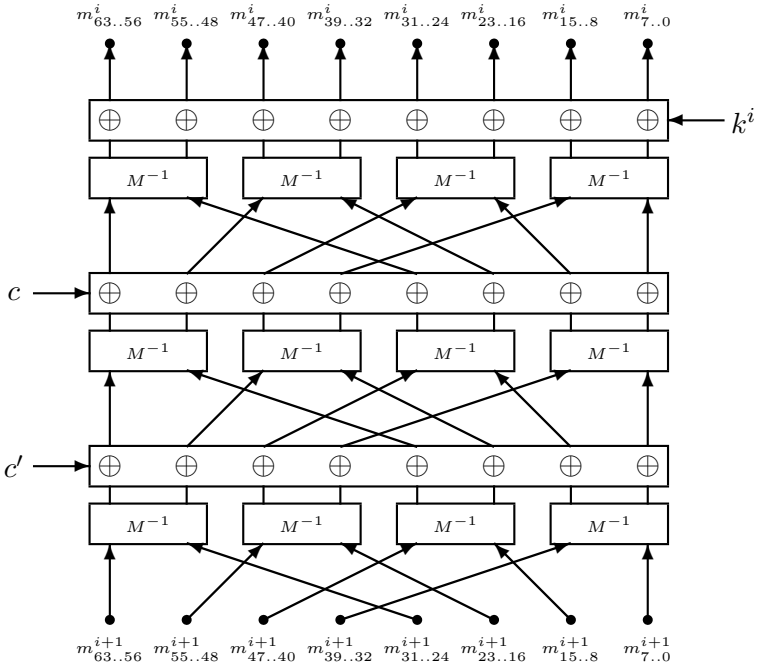


FIGURE G.6 – One decryption round

G.1.5 Test Values

As an example we encrypt the plaintext $\texttt{0123456789abcdef}_x$ with the secret key $\texttt{0123456789abcdeffedcba9876543210}_x$. The subkeys sequence is

$$\begin{aligned}
k^{-2} &= \texttt{fedcba9876543210}_x \\
k^{-1} &= \texttt{0123456789abcdef}_x \\
k^{0} &= \texttt{45fd137a4edf9ec4}_x \\
k^{1} &= \texttt{1dd43f03e6f7564c}_x \\
k^{2} &= \texttt{ebe26756de9937c7}_x \\
k^{3} &= \texttt{961704e945bad4fb}_x \\
k^{4} &= \texttt{0b60dfe9eff473d4}_x \\
k^{5} &= \texttt{76d3e7cf52c466cf}_x \\
k^{6} &= \texttt{75ec8cef767d3a0d}_x \\
k^{7} &= \texttt{82da3337b598fd6d}_x \\
k^{8} &= \texttt{fbd820da8dc8af8c}_x
\end{aligned}$$

For instance, the first generated subkey $k^0 = \texttt{45fd137a4edf9ec4}_x$ is

$$\begin{aligned}
k^0 &= k^{-2} \oplus T(P^8(k^{-1} \oplus \texttt{290d61409ceb9e8f}_x)) \\
&= k^{-2} \oplus T(\texttt{b711fa89ae0394e4}_x) \\
&= k^{-2} \oplus \texttt{bb21a9e2388bacd4}_x.
\end{aligned}$$

The messages which enter into each round are

$$\begin{aligned}
m^0 &= \texttt{0123456789abcdef}_x \\
m^1 &= \texttt{c3feb96c0cf4b649}_x \\
m^2 &= \texttt{3f54e0c8e61a84d1}_x \\
m^3 &= \texttt{b15cb4af3786976e}_x \\
m^4 &= \texttt{76c122b7a562ac45}_x \\
m^5 &= \texttt{21300b6ccfaa08d8}_x \\
m^6 &= \texttt{99b8d8ab9034ec9a}_x \\
m^7 &= \texttt{a2245ba3697445d2}_x
\end{aligned}$$

and the ciphertext is $\texttt{88fddfbe954479d7}_x$. In the first round, the message m^0 is transformed though three layers into m^1. The intermediate results between the layers are $\texttt{d85c19785690b0e3}_x$ and $\texttt{0f4bfb9e2f8ac7e2}_x$. For instance, in the first layer we take m^0, xor it with k^0, apply M, permute the bytes and get $\texttt{d85c19785690b0e3}_x$.

As an implementation test, we mention that if we iterate one million times the encryption on the all-zero bitstring with the previous key, we obtain the final ciphertext $\texttt{fd5c9c6889784b1c}_x$.

G.2 Design Arguments

The Fast Fourier Transform used in the round-encryption function E has been used in several cryptographic designs including Schnorr's FFT-Hashing [141], FFT Hash II [143], Schnorr and Vaudenay's Parallel FFT-Hashing [145], and Massey's SAFER [87, 88]. This graph has been proved to have very good diffusion properties when done twice (see [146, 147, 163]).

The M structure implements a *multipermutation* as defined by Schnorr and Vaudenay (see [146, 162, 163]). In this case, it means that M is a permutation over the set of all 16-bit strings, and that fixing any of the two 8-bit inputs arbitrarily makes both 8-bit outputs be permutations of the other one. This is due to a very particular property of φ, namely that both φ and $x \mapsto \varphi(x) \oplus R_l(x)$ (which is in fact φ') are permutations. Actually, φ and φ' are linear involutions.

Those properties make E be what we call a *mixing function*, *i.e.* such that if we arbitrarily fix seven of the eight inputs, all outputs are permutation of the remaining free input. This performs a good diffusion.

The best general attack methods on block ciphers have been introduced by Gilbert, Chassé, Tardy-Corfdir, Biham, Shamir and Matsui (see [54, 158, 29, 30, 89, 91, 53]). They are now known as differential and linear cryptanalysis. We know study how CS-CIPHER has been protected against it.

The permutation P has been chosen to be an nonlinear involution in the sense that both differential and linear cryptanalysis are hard. Nonlinearity has one measure corresponding to differential cryptanalysis (which has been defined by Nyberg [110]) and one measure corresponding to linear cryptanalysis (which has been defined by Chabaud and Vaudenay [35]). Here we use the formalism introduced by Matsui [92]:

$$\begin{aligned} \mathrm{DP}_{\max}(f) &= \max_{a \neq 0, b} \Pr_{X \text{ uniform}}[f(X \oplus a) \oplus f(X) = b] \\ \mathrm{LP}_{\max}(f) &= \max_{a,b \neq 0} \left(2 \Pr_{X \text{ uniform}}[X \cdot a = f(X) \cdot b] - 1\right)^2 . \end{aligned}$$

The functions f and g are such that $\mathrm{DP}_{\max}(f) \leq 2^{-2}$ and $\mathrm{LP}_{\max}(f) \leq 2^{-2}$. If the Theorem of Aoki and Ohta [18] (which generalizes the Theorem of Nyberg and Knudsen [115]) were applicable in this setting, we would then obtain $\mathrm{DP}_{\max}(P) \leq 2^{-4}$ and $\mathrm{LP}_{\max}(P) \leq 2^{-4}$. Both properties are however

still satisfied as the experiment shows. From [110, 35] it is known that for any function f on the set of all n-bit strings we have $\mathrm{DP}_{\max}(f) \geq 2^{1-n}$ and $\mathrm{LP}_{\max}(f) \geq 2^{1-n}$, but it is conjectured that 2^{2-n} is a better bound for even n (see Dobbertin [49] for instance). So our functions are reasonably nonlinear. Since it is well known that the heuristic complexity of differential or linear cryptanalysis is greater than the inverse of the product of the $\mathrm{DP}_{\max}$ or $\mathrm{LP}_{\max}$ of all active P boxes (see for instance Heys and Tavares [64]), having mixing functions makes at least five P box per round to be active, so no four rounds of CS-Cipher have any efficient differential or linear characteristic.

G.3 Implementation

In any kind of implementation, the key scheduling scheme is assumed to be precomputed. This part of CS-Cipher has not be designed to have special implementation optimization. The authors believe that every time one changes the secret key, one has to perform expensive computations (such as asymmetric cryptography, key exchange protocol or key transfer protocols) so optimizing the precomputation of the subkey sequence is meaningless. In the following Sections we only discuss implementation of the encryption (or decryption) scheme.

G.3.1 VLSI Implementation

CS-Cipher is highly optimized for VLSI implementations. It may be noticed that the g function has been designed to get a friendly boolean circuit implementation. Actually, Figure G.7 illustrates a cheap nand-circuit with depth 4 and only 16 nand gates.

We propose two possible easy implementations. In the first one, we really implement one third of a single round encryption. It has two 64-bit input registers and one 64-bit output register. It is easy to see that an encryption can be performed by iterating this circuit 24 times and loading the sequence $k^0, c, c', k^1, c, c', \ldots$ Straightforward estimates shows this circuit requires 1216 nand-gates with depth 26. This implementation can be added in any microprocessor within less than $1\mathrm{mm}^2$ in order to get a simple microcoded encryption instruction. One 30MHz-clock cycle is far enough to compute one layer, thus one 64-bit encryption requires 24 clock cycles, which leads to a 73Mbps, which is quite fast for such a cheap technology.

The second implementation consists in making a dedicated chip which consists of 24 times the previous circuit in a pipeline architecture. We estimate we need $15\mathrm{mm}^2$ in order to implement a 30000 nand-gate circuit which

layer4 :	$g_0 = g_4 n g_5$	$g_1 = g_6 n g_7$	$g_2 = g_8 n g_9$	$g_3 = g_{10} n g_{11}$
layer3 :	$g_5 = g_6 n g_{14}$	$g_8 = g_{17} n g_4$	$g_9 = g_7 n g_{14}$	$g_{10} = g_6 n g_{16}$
layer2 :	$g_6 = g_{14} n g_{12}$	$g_7 = g_{15} n g_{16}$	$g_{11} = g_{13} n g_4$	$g_{17} = g_{15} n g_{19}$
layer1 :	$g_4 = g_{12} n g_{13}$	$g_{14} = g_{18} n g_{18}$	$g_{15} = g_{18} n g_{12}$	$g_{19} = g_{16} n g_{13}$
layer0 :	g_{12}	g_{13}	g_{16}	g_{18}

Input : $g_{16} g_{13} g_{18} g_{12}$
Output : $g_0 g_1 g_2 g_3$

FIGURE G.7 – Implementation of g

performs a 64-bit encryption within one 30MHz-clock cycle, which leads to an encryption rate of 2Gbps. This can be used to encrypt ATM network communications or PCI bus.

Those results can be compared to MISTY which has been implemented by Mitsubishi. In Matsui [93], this chip is specified to require 65000 gates, working at 14MHz and encrypting at 450Mbps.

G.3.2 Software Implementation on Modern Microprocessors

A straightforward non-optimized implementation of CS-CIPHER in standard C on a Pentium 133MHz (see Appendix) gives an encryption rate of 2.1Mbps which is reasonably fast compared to similar implementations of DES.

Another (non-optimized) implementation in assembly code enables the Pentium to perform a 64-bit encryption within 973 clock cycles, which leads to 8.34Mbps working at 133MHz.

An evaluation similar to the VLSI-implementation estimates shows that the number of "usual" boolean gate (xor, and, or not) required to implement 64 parallel 64-bit encryptions using Biham's bit-slice trick on a 64-bit microprocessor is 11968, which is substantially less than Biham's implementation of DES which requires about 16000 instructions (see [26]). Therefore, if we use a 300MHz Alpha microprocessor which requires .5cycles per instructions (as in [26]), we obtain an encryption rate of about 196Mbps.

platform	clock frequency	encryption rate	note
VLSI 1216nand 1mm^2	30MHz	73Mbps	estimate
VLSI 30000nand 15mm^2	30MHz	2Gbps	estimate
standard C 32bits	133MHz	2Mbps	see Appendix
bit slice (Pentium)	133MHz	11Mbps	estimate
bit slice (Alpha)	300MHz	196Mbps	estimate
Pentium assembly code	133MHz	8Mbps	non-optimized
6805 assembly code	4MHz	20Kbps	non-optimized

Figure G.8 – Implementations of CS-Cipher

G.3.3 Software Implementation on 8-Bit Microprocessors

An implementation has been done for a cheap smart card platform. A compact 6805 assembly code of roughly 500 bytes can encrypt a 64-bit string in its buffer RAM by using only 6 extra byte-registers within 12633 clock cycles. This means that a cheap smart card working at 4MHz can encrypt within 3,16ms (*i.e.* at a 19,8Kbps rate), which is better than optimized implementations of DES[2]. This implementation of CS-Cipher can still be optimized.

G.4 Conclusion

CS-Cipher has been shown to offer quite fast encryption rates on several kinds of platforms, which is suitable for telecommunication applications. Figure G.8 summerizes the implementation results. Its security is based on heuristic arguments.

All attacks are welcome...

Acknowledgements

We wish to thank the Compagnie des Signaux for having initiated and supported this work.

Appendix

Here is a sample implementation of the heart of CS-Cipher. The procedure takes plaintext block `m` and a precomputed subkey sequence `k` (as a 9×8

bytes array). This program is highly optimizable.

```
typedef unsigned char uint8;
#define CSC_C00 0xb7
#define CSC_C01 0xe1
#define CSC_C02 0x51
#define CSC_C03 0x62
#define CSC_C04 0x8a
#define CSC_C05 0xed
#define CSC_C06 0x2a
#define CSC_C07 0x6a
#define CSC_C10 0xbf
#define CSC_C11 0x71
#define CSC_C12 0x58
#define CSC_C13 0x80
#define CSC_C14 0x9c
#define CSC_C15 0xf4
#define CSC_C16 0xf3
#define CSC_C17 0xc7
uint8 tbp[256]={
  0x29,0x0d,0x61,0x40,0x9c,0xeb,0x9e,0x8f,
  0x1f,0x85,0x5f,0x58,0x5b,0x01,0x39,0x86,
  0x97,0x2e,0xd7,0xd6,0x35,0xae,0x17,0x16,
  0x21,0xb6,0x69,0x4e,0xa5,0x72,0x87,0x08,
  0x3c,0x18,0xe6,0xe7,0xfa,0xad,0xb8,0x89,
  0xb7,0x00,0xf7,0x6f,0x73,0x84,0x11,0x63,
  0x3f,0x96,0x7f,0x6e,0xbf,0x14,0x9d,0xac,
  0xa4,0x0e,0x7e,0xf6,0x20,0x4a,0x62,0x30,
  0x03,0xc5,0x4b,0x5a,0x46,0xa3,0x44,0x65,
  0x7d,0x4d,0x3d,0x42,0x79,0x49,0x1b,0x5c,
  0xf5,0x6c,0xb5,0x94,0x54,0xff,0x56,0x57,
  0x0b,0xf4,0x43,0x0c,0x4f,0x70,0x6d,0x0a,
  0xe4,0x02,0x3e,0x2f,0xa2,0x47,0xe0,0xc1,
  0xd5,0x1a,0x95,0xa7,0x51,0x5e,0x33,0x2b,
  0x5d,0xd4,0x1d,0x2c,0xee,0x75,0xec,0xdd,
  0x7c,0x4c,0xa6,0xb4,0x78,0x48,0x3a,0x32,
  0x98,0xaf,0xc0,0xe1,0x2d,0x09,0x0f,0x1e,
  0xb9,0x27,0x8a,0xe9,0xbd,0xe3,0x9f,0x07,
  0xb1,0xea,0x92,0x93,0x53,0x6a,0x31,0x10,
  0x80,0xf2,0xd8,0x9b,0x04,0x36,0x06,0x8e,
  0xbe,0xa9,0x64,0x45,0x38,0x1c,0x7a,0x6b,
  0xf3,0xa1,0xf0,0xcd,0x37,0x25,0x15,0x81,
  0xfb,0x90,0xe8,0xd9,0x7b,0x52,0x19,0x28,
  0x26,0x88,0xfc,0xd1,0xe2,0x8c,0xa0,0x34,
  0x82,0x67,0xda,0xcb,0xc7,0x41,0xe5,0xc4,
  0xc8,0xef,0xdb,0xc3,0xcc,0xab,0xce,0xed,
  0xd0,0xbb,0xd3,0xd2,0x71,0x68,0x13,0x12,
  0x9a,0xb3,0xc2,0xca,0xde,0x77,0xdc,0xdf,
  0x66,0x83,0xbc,0x8d,0x60,0xc6,0x22,0x23,
  0xb2,0x8b,0x91,0x05,0x76,0xcf,0x74,0xc9,
  0xaa,0xf1,0x99,0xa8,0x59,0x50,0x3b,0x2a,
  0xfe,0xf9,0x24,0xb0,0xba,0xfd,0xf8,0x55,
};
void enc_csc(uint8 m[8],uint8* k) {
  uint8 tmpx,tmprx,tmpy;
  int i;
  #define APPLY_M(cl,cr,adl,adr) \
    tmpx=m[adl]^cl; \
    tmprx=(tmpx<<1)^(tmpx>>7); \
    tmpy=m[adr]^cr; \
    m[adl]=tbp[(tmprx&0x55)^tmpx^tmpy]; \
    m[adr]=tbp[tmprx^tmpy];
```

```
  for(i=0;i<8;i++,k+=8) {
    APPLY_M(k[0],k[1],0,1)
    APPLY_M(k[2],k[3],2,3)
    APPLY_M(k[4],k[5],4,5)
    APPLY_M(k[6],k[7],6,7)
    APPLY_M(CSC_C00,CSC_C01,0,2)
    APPLY_M(CSC_C02,CSC_C03,4,6)
    APPLY_M(CSC_C04,CSC_C05,1,3)
    APPLY_M(CSC_C06,CSC_C07,5,7)
    APPLY_M(CSC_C10,CSC_C11,0,4)
    APPLY_M(CSC_C12,CSC_C13,1,5)
    APPLY_M(CSC_C14,CSC_C15,2,6)
    APPLY_M(CSC_C16,CSC_C17,3,7)
  }
  for(i=0;i<8;i++) m[i]^=k[i];
}
```

Annexe H

Decorrelated Fast Cipher: an AES Candidate

[*Cet article écrit par*

Henri Gilbert, Marc Girault, Philippe Hoogvorst,
Fabrice Noilhan, Thomas Pornin, Guillaume Poupard,
Jacques Stern *et* Serge Vaudenay

a été publié dans les actes du séminaire Advanced Encryption Standard Workshop I *en 1998 [55]. Une version plus complète fait partie du dossier soumis en réponse à l'appel d'offre AES du gouvernement américain [56]. Une présentation a également été faite au colloque Cardis 98 [127].*]

[**Erratum.** KC and KD have been flipped in the definition of EES p. 210. Equation (H.22) should read

$$\mathrm{EES} = \mathrm{RT}(0)|\mathrm{RT}(1)|\ldots|\mathrm{RT}(63)|\mathrm{KD}|\mathrm{KC}.$$

Similarly, the last two lines of EES in Equation (H.24) should read

$$\texttt{78d56ced 94640d6e f0d3d37b e67008e1 86d1bf27 5b9b241d}_x$$
$$\texttt{eb64749a}_x$$

which is the correct expansion of e.]

Abstract. This report presents a response to the call for candidates issued by the National Institute for Standards and Technologies (the Advanced Encryption Standard project). The proposed candidate —

called DFC as for "Decorrelated Fast Cipher" — is based on the recent decorrelation technique. This provides provable security against several classes of attacks which include Differential Cryptanalysis and Linear Cryptanalysis.

Digital criminality is nowadays a big threat for the electronic marketplace. For this reason, cryptography provides various algorithms based on a heart cryptographic primitive: encryption. The Digital Encryption Standard (DES) has been developed by IBM™ for the US Department of Commerce in the seventies for this purpose, but its secret-key length (56 bits) provides no sufficient security at this time, so this standard is now over.

So far, real-life encryption algorithms used to have an empirical-based security: they were designed from an intricate substitution-permutation network and believed to be secure until someone published an attack on them. In parallel, research yielded several general attacks strategies, namely Biham and Shamir's "differential cryptanalysis" and Matsui's "linear cryptanalysis" (both are particular cases of the more general "iterated attacks of order 2"), which provided a better understanding on how to manage with security arguments.

The laboratory of computer sciences of the *Ecole Normale Supérieure*, associated with the *Centre National pour la Recherche Scientifique* (CNRS), has recently developed a technique for making new encryption algorithms with a **provable security** against any iterated attacks of a fixed order (*e.g.* of order 2). Several properties of this technique — known as **decorrelation** — have been presented at international research conferences. In this extended abstract, we present a candidate for the "Advanced Encryption Standard" process of the US Department of Commerce, and which is based on decorrelation. We call it DFC as for "Decorrelated Fast Cipher".

DFC enables to encrypt any digital information with a key of length up to 256 bits. It has been implemented on various computer platforms with the following benchmarks.

microprocessor	cycles-per-bit	clock-frequency	bits-per-second
AXP™	4.36	600MHz	137.6Mbps
Pentium™	5.89	200MHz	34.0Mbps
SPARC™	6.27	170MHz	27.1Mbps

In addition, it has been implemented on a cheap smart card based on the Motorola™ 6805 microprocessor for which one block encryption requires 9.80ms. All these experiments yield a speed rate greater than all commercial implementations of DES, and with a much higher security.

Provable security is an important added value for cryptographic algorithms and is currently a hot topic in international conferences. The decorrelation technique is a part of this program.

H.1 Notations

All objects are bit strings or integers. Bit strings are represented in hexadecimal notations. For instance, $\texttt{d43}_x$ denotes the bit string `110101000011`. Integers are represented in standard decimal notations. The notations used to manipulate them are as follows.

$\bar{s}$	convert bit string into an integer.
$\lvert x\rvert_\ell$	convert integer x into a bit string of length ℓ.
$s\lvert s'$	concatenation of two strings.
$\mathrm{trunc}_n(s)$	truncate a bit string to its n leftmost bits.
$s \oplus s'$	bitwise XOR
$s \wedge s'$	bitwise *and*
$\neg s$	bitwise negation.
$+, \times, \mathrm{mod}$	natural arithmetic operations over the integers.

For instance, $\overline{\texttt{d43}_x} = 3395$ and $\lvert 3395\rvert_{12} = \texttt{d43}_x$.

H.2 High Level Overview

The encryption function DFC_K operates on 128-bit message blocks by means of a secret key K of arbitrary length, up to 256 bits. The corresponding decryption function is DFC_K^{-1} and operates on 128-bit message blocks. Encryption of arbitrary-length messages is performed through standard modes of operation.

The secret key K is first turned into a 1024-bit "Expanded Key" EK through an "Expanding Function" EF, *i.e.* $\mathrm{EK} = \mathrm{EF}(K)$. As explained in Section H.5, the EF function performs a 4-round Feistel scheme (see Feistel [51]). The encryption itself performs a similar 8-round Feistel scheme. Each round uses the "Round Function" RF. This function maps a 64-bit string onto a 64-bit string by using one 128-bit string parameter. It is defined in Section H.3.

Given a 128-bit plaintext block PT, we split it into two 64-bit halves R_0 and R_1 so that $\mathrm{PT} = R_0|R_1$. Given the 1024-bit expanded key EK, we split it into eight 128-bit strings

$$\mathrm{EK} = \mathrm{RK}_1|\mathrm{RK}_2|\ldots|\mathrm{RK}_8 \tag{H.1}$$

where RK_i is the ith "Round Key".

We build a sequence $R_0, \ldots, R_9$ by the Equation

$$R_{i+1} = \mathrm{RF}_{\mathrm{RK}_i}(R_i) \oplus R_{i-1}. \quad (i = 1, \ldots, 8) \tag{H.2}$$

We then set $\mathrm{CT} = \mathrm{DFC}_K(\mathrm{PT}) = R_9|R_8$ (see Fig. H.1).

More generally, given a bitstring s of length multiple of 128, say $128r$, we can split it into r 128-bit strings

$$s = p_1|p_2|\ldots|p_r.$$

From s we define a permutation Enc_s on the set of 128-bit strings which comes from an r-round Feistel scheme. For any 128-bit string m which is split into two 64-bit halves x_0 and x_1 so that $m = x_0|x_1$. We build a sequence $x_0, \ldots, x_{r+1}$ by the Equation

$$x_{i+1} = \mathrm{RF}_{p_i}(x_i) \oplus x_{i-1} \quad (i = 1, \ldots, r) \tag{H.3}$$

and we define $\mathrm{Enc}_s(m) = x_{r+1}|x_r$. The DFC_K encryption function is thus obtained as

$$\mathrm{DFC}_K = \mathrm{Enc}_{\mathrm{EF}(K)} \tag{H.4}$$

(hence an 8-round Feistel Cipher). The EF function uses a 4-round version defined with Enc.

Obviously, we have $\mathrm{DFC}_K^{-1} = \mathrm{Enc}_{\mathrm{revEK}}$ where

$$\mathrm{revEK} = \mathrm{RK}_8|\mathrm{RK}_7|\ldots|\mathrm{RK}_1. \tag{H.5}$$

H.3 The RF Function

The RF function (as for "Round Function") is fed with one 128-bit parameter, or equivalently two 64-bit parameters: an "a-parameter" and a "b-parameter". It processes a 64-bit input x and outputs a 64-bit string. We define

$$\mathrm{RF}_{a|b}(x) = \mathrm{CP}\left(\left|\left(\left(\bar{a} \times \bar{x} + \bar{b}\right) \bmod (2^{64} + 13)\right) \bmod 2^{64}\right|_{64}\right) \tag{H.6}$$

where CP is a permutation over the set of all 64-bit strings (which appears in Section H.4). This construction is the "pairwise decorrelation module".

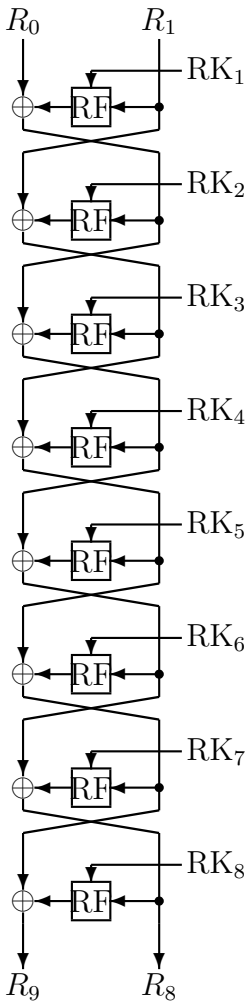


Figure H.1 – An 8-Round Feistel Cipher.

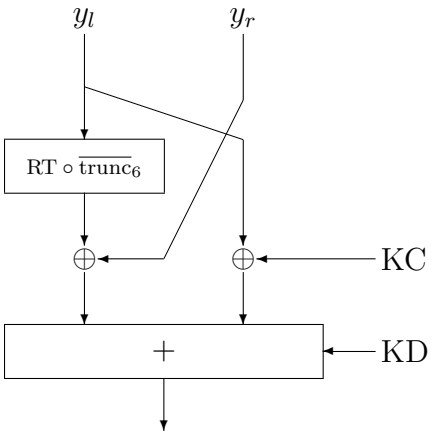


FIGURE H.2 – The CP Permutation.

H.4 The CP Permutation

The CP permutation (as for "Confusion Permutation") uses a look-up table RT (as for "Round Table") which takes a 6-bit integer as input and provides a 32-bit string output.

Let $y = y_l|y_r$ be the input of CP where y_l and y_r are two 32-bit strings. We define

$$\text{CP}(y) = \left|\overline{(y_r \oplus \text{RT}(\overline{\text{trunc}_6(y_l)}))|(y_l \oplus \text{KC})} + \overline{\text{KD}} \bmod 2^{64}\right|_{64} \tag{H.7}$$

where KC is a 32-bit constant string, and KD is a 64-bit constant string. Permutation CP is depicted on Fig. H.2.

The constants $\text{RT}(0), \ldots, \text{RT}(63)$, KC and KD will be set in Section H.6.

H.5 Key Scheduling Algorithm

In order to generate a sequence $\text{RK}_1, \text{RK}_2, \ldots, \text{RK}_8$ from a given key K represented as a bit string of length at most 256, we use the following algorithm. We first pad K with a constant pattern KS in order to make a 256-bit "Padded Key" string by

$$\text{PK} = \text{trunc}_{256}(K|\text{KS}). \tag{H.8}$$

If K is of length 128, we can observe that only the first 128 bits of KS are used. We define KS of length 256 in order to allow any key size from 0 to 256.

Then we cut PK into eight 32-bit strings $\mathrm{PK}_1, \ldots, \mathrm{PK}_8$ such that $\mathrm{PK} = \mathrm{PK}_1 | \ldots | \mathrm{PK}_8$. We define

$$\begin{aligned} \mathrm{OAP}_1 &= \mathrm{PK}_1 | \mathrm{PK}_8 && \text{(H.9)} \\ \mathrm{OBP}_1 &= \mathrm{PK}_5 | \mathrm{PK}_4 && \text{(H.10)} \\ \mathrm{EAP}_1 &= \mathrm{PK}_2 | \mathrm{PK}_7 && \text{(H.11)} \\ \mathrm{EBP}_1 &= \mathrm{PK}_6 | \mathrm{PK}_3. && \text{(H.12)} \end{aligned}$$

We also define

$$\begin{aligned} \mathrm{OAP}_i &= \mathrm{OAP}_1 \oplus \mathrm{KA}_i && \text{(H.13)} \\ \mathrm{OBP}_i &= \mathrm{OBP}_1 \oplus \mathrm{KB}_i && \text{(H.14)} \\ \mathrm{EAP}_i &= \mathrm{EAP}_1 \oplus \mathrm{KA}_i && \text{(H.15)} \\ \mathrm{EBP}_i &= \mathrm{EBP}_1 \oplus \mathrm{KB}_i && \text{(H.16)} \end{aligned}$$

for $i = 2, 3, 4$ (where KA_i and KB_i are fixed constants defined in Section H.6). The names of the variables come from "Odd a-Parameter", "Odd b-Parameter", "Even a-Parameter", and "Even b-Parameter" respectively, which will become clearer below.

We define

$$\mathrm{EF}_1(K) = \mathrm{OAP}_1 | \mathrm{OBP}_1 | \ldots | \mathrm{OAP}_4 | \mathrm{OBP}_4. \tag{H.17}$$

It defines a four-round permutation which is $\mathrm{Enc}_{\mathrm{EF}_1(K)}$. Similarly,

$$\mathrm{EF}_2(K) = \mathrm{EAP}_1 | \mathrm{EBP}_1 | \ldots | \mathrm{EAP}_4 | \mathrm{EBP}_4 \tag{H.18}$$

defines a four-round encryption function $\mathrm{Enc}_{\mathrm{EF}_2(K)}$.

The $\mathrm{Enc}_{\mathrm{EF}_1(K)}$ and $\mathrm{Enc}_{\mathrm{EF}_2(K)}$ enables to define the RK sequence. Namely, we let $\mathrm{RK}_0 = |0|_{128}$ and

$$\mathrm{RK}_i = \begin{cases} \mathrm{Enc}_{\mathrm{EF}_1(K)}(\mathrm{RK}_{i-1}) & \text{if } i \text{ is odd} \\ \mathrm{Enc}_{\mathrm{EF}_2(K)}(\mathrm{RK}_{i-1}) & \text{if } i \text{ is even.} \end{cases} \tag{H.19}$$

Finally we have

$$\mathrm{EF}(K) = \mathrm{RK}_1 | \mathrm{RK}_2 | \ldots | \mathrm{RK}_8. \tag{H.20}$$

H.6 On Defining the Constants

The previously defined algorithm depends on several constants:

- 64 constants $\mathrm{RT}(|0|_6), \ldots, \mathrm{RT}(|63|_6)$ of 32 bits,

- one 64-bit constant KD,
- one 32-bit constant KC,
- three 64-bit constants $\mathrm{KA}_2, \mathrm{KA}_3, \mathrm{KA}_4$,
- three 64-bit constants $\mathrm{KB}_2, \mathrm{KB}_3, \mathrm{KB}_4$,
- one 256-bit constant KS.

In order to convince that this design hides no trap-door, we choose the constants from the hexadecimal expansion of the mathematical e constant

$$e = \sum_{n=0}^{\infty} \frac{1}{n!} = 2.\texttt{b7e151628aed2a6abf7158}_x \ldots \tag{H.21}$$

If EES is the "e Expansion String" of the first 2144 bits of this expansion (after the decimal point), we define

$$\mathrm{EES} = \mathrm{RT}(0)|\mathrm{RT}(1)|\ldots|\mathrm{RT}(63)|\mathrm{KC}|\mathrm{KD}. \tag{H.22}$$

In addition we define

$$\mathrm{trunc}_{640}(\mathrm{EES}) = \mathrm{KA}_2|\mathrm{KA}_3|\mathrm{KA}_4|\mathrm{KB}_2|\mathrm{KB}_3|\mathrm{KB}_4|\mathrm{KS}. \tag{H.23}$$

Here is the EES string.

$$\begin{array}{l}
\texttt{b7e15162 8aed2a6a bf715880 9cf4f3c7 62e7160f 38b4da56}_x \\
\texttt{a784d904 5190cfef 324e7738 926cfbe5 f4bf8d8d 8c31d763}_x \\
\texttt{da06c80a bb1185eb 4f7c7b57 57f59584 90cfd47d 7c19bb42}_x \\
\texttt{158d9554 f7b46bce d55c4d79 fd5f24d6 613c31c3 839a2ddf}_x \\
\texttt{8a9a276b cfbfa1c8 77c56284 dab79cd4 c2b3293d 20e9e5ea}_x \\
\texttt{f02ac60a cc93ed87 4422a52e cb238fee e5ab6add 835fd1a0}_x \\
\texttt{753d0a8f 78e537d2 b95bb79d 8dcaec64 2c1e9f23 b829b5c2}_x \\
\texttt{780bf387 37df8bb3 00d01334 a0d0bd86 45cbfa73 a6160ffe}_x \\
\texttt{393c48cb bbca060f 0ff8ec6d 31beb5cc eed7f2f0 bb088017}_x \\
\texttt{163bc60d f45a0ecb 1bcd289b 06cbbfea 21ad08e1 847f3f73}_x \\
\texttt{78d56ced 94640d6e f0d3d37b e67008e1 eb64749a 86d1bf27}_x \\
\texttt{5b9b241d}_x
\end{array} \tag{H.24}$$

H.7 Security Results

The design construction is based on decorrelation techniques (see [173, 174, 177]). From the decorrelation theory we know that a six-round Feistel cipher which uses RF with independent subkeys has a pairwise decorrelation distance less than $0{,}821.2^{-113}$ to the Perfect Cipher. We can thus give lower bounds on the complexity of differential cryptanalysis, linear cryptanalysis and general iterated attacks of order 1 which achieve an advantage at least 10%.

attack	differential	linear	iterated
complexity lower bound	2^{110}	2^{92}	2^{48}

These are attacks against a six-round encryption function when assuming that EK has a uniform distribution. It is applicable to DFC with the following assumption.

> "We cannot distinguish $\mathrm{Enc}_{\mathrm{EF}_1(K)}$ from the a truly random permutation within an advantage greater than 1%, with only 4 chosen plaintexts and a limited budget of US$\$1,000,000,000$."

(Limiting the budget gives an upper bound on the computation cost.)

These results suggest that the key should not be used more than 2^{48} times *i.e.* that we should not encrypt 4096TB with the same key. We believe that this restricts no practical application.

We also (pessimisticly) investigated the complexity of exhaustive search by extrapolating the technology improvements. We obtained the following rationales.

key length	80	128	192	256
computation lower bound (in years)	21.7	93.7	126.4	190.4

In our full report we also outlined that the DFC algorithm is weak when reduced to four rounds. We believe the decorrelation technique makes enough avalanche effect so that eight rounds provide a sufficient security.

H.8 Conclusion

We have proposed a dedicated block cipher algorithm which is faster than DES and hopefully more secure than triple-DES. In addition we provided

proofs of security against some classes of general simple attacks which includes differential and linear cryptanalysis. This result is based on the decorrelation theory. We believe that this cipher is also "naturally" secure against more complicated attacks since our design introduced no special algebraic property. We believe that the best attack is still exhaustive search which is limited by the implementation speed (decreased by a factor of 5 due to the key scheduling algorithm). We (very pessimisticly) forecast that one need at least several decades to search a 80-bit key, which makes it safe until the Advanced Encryption Standard expires.

Our algorithm accepts 128-bit message blocks and any key size from 0 to 256. It can be adapted into a 64-bit variant (with a key size up to 128) as shown in the full report. We believe that it can be adapted to any other cryptographic primitive such as stream cipher, hash function, MAC algorithm.

Our algorithm can be implemented on traditional personal computers, as well as on cheap smart cards. We believe that it can be implemented in any other digital environment.

In conclusion we recommend this encryption algorithm as a candidate to the Advanced Encryption Standard process.

Annexe I

Cryptanalysis of the Chor-Rivest Cryptosystem

[*Cet article de* Serge Vaudenay *a été publié dans les actes du colloque Crypto 98 [178] et soumis au Journal of Cryptology.*]

Abstract. Knapsack-based cryptosystems used to be popular in the beginning of public key cryptography before being all broken, all but the Chor-Rivest cryptosystem. In this paper, we show how to break this one with its suggested parameters: $\mathrm{GF}(p^{24})$ and $\mathrm{GF}(256^{25})$. We also give direction on possible extensions of our attack.

Recent interests about cryptosystems based on knapsacks or lattice reduction problems unearthed the problem of their security. So far, the Chor-Rivest was the only unbroken cryptosystem based on the subset sum problem [36, 37]. In this paper, we present a new attack on it which definitely breaks the system for all the proposed parameters in Chor-Rivest's final paper [37]. We also give directions to break the general problem, and related cryptosystems such as Lenstra's Powerline cryptosystem [84].

I.1 The Chor-Rivest Cryptosystem

We let $q = p^h$ be a power-prime (for a practical example, let $p = 197$ and $h = 24$). We consider the finite field $\mathrm{GF}(q)$ and we assume that its representation is public (*i.e.* there is a public h-degreed polynomial $P(x)$ irreducible on $\mathrm{GF}(p)$ and elements of $\mathrm{GF}(q)$ are polynomials modulo $P(x)$). We also consider a public numbering α of the subfield $\mathrm{GF}(p)$, *i.e.* $\{\alpha_0, \ldots, \alpha_{p-1}\} = \mathrm{GF}(p) \subseteq \mathrm{GF}(q)$.

Secret keys consist of

- an element $t \in \mathrm{GF}(q)$ with algebraic degree h
- a generator g of $\mathrm{GF}(q)^*$
- an integer $d \in \mathbf{Z}_{q-1}$
- a permutation π of $\{0, \ldots, p-1\}$

Public keys consist of all

$$c_i = d + \log_g(t + \alpha_{\pi(i)}) \bmod q - 1$$

for $i = 0, \ldots, p-1$. For this reason, the public parameters must be chosen such that the discrete logarithm is easy to calculate in $\mathrm{GF}(q)$. In the final paper, the authors suggested to use a relatively small prime power p and a smooth power h, *i.e.* an integer with only small factors so that we can apply the Pohlig-Hellman algorithm [123].[1] Suggested parameters corresponds to the fields $\mathrm{GF}(197^{24})$, $\mathrm{GF}(211^{24})$, $\mathrm{GF}(243^{24})$, and $\mathrm{GF}(256^{25})$.

The Chor-Rivest cryptosystem works over a message space which consists of all p-bit strings with Hamming weight h. This means that the message to be encrypted must first be encoded as a bitstring $m = [m_0 \ldots m_{p-1}]$ such that $m_0 + \ldots + m_{p-1} = h$. The ciphertext space is $\mathbf{Z}_{q-1}$ and we have

$$E(m) = m_0 c_0 + \ldots + m_{p-1} c_{p-1} \bmod q - 1.$$

To decrypt the ciphertext $E(m)$, we compute

$$p(t) = g^{E(m) - hd}$$

as a polynomial in term of t over $\mathrm{GF}(p)$ with degree at most $h-1$, which must be equal to

$$\prod_{m_i=1} (t + \alpha_{\pi(i)})$$

in $\mathrm{GF}(q)$. Thus, if we consider $\mu(x) + p(x)$ where $\mu(x)$ is the minimal polynomial of t, we must obtain the formal polynomial

$$\prod_{m_i=1} (x + \alpha_{\pi(i)})$$

[1] This algorithm with Shanks' baby step giant step trick has a complexity of $O(h^3\sqrt{B}\log p)$ simple $\mathrm{GF}(p)$-operations for computing one c_i where B is the largest prime factor of $p^h - 1$. (See Koblitz [76].) Since $p^r - 1$ is a factor of $p^h - 1$ when r is a factor of h, B is likely to be small when h only has small prime factors.

whose factorization leads to m.

Although the public key generation relies on intricate finite fields computations, the decryption problem is based on the traditional subset sum problem (also more familiarly called *knapsack problem*): given a set of pieces $c_0, \ldots, c_{p-1}$ and a target $E(m)$, find a subset of pieces so that its sum is $E(m)$. This problem is known to be hard, but the cryptosystem hides a trapdoor which enables the legitimate user to decrypt. This modifies the genericity of the problem and the security is thus open.

I.2 Previous Work

The Merkle-Hellman cryptosystem was the first subset-sum-based cryptosystem [103]. Although the underlying problem is NP-complete, it has surprisingly been broken by Shamir [148]. Later, many other variants have been shown insecure for any practical parameters by lattice reduction techniques (see [68] for instance). Actually, subset-sum problems can be characterized by the density parameter which is (with our notations) the ratio $d = p/\log_2 q$. When the density is far from 1 (which was the case of most of cryptosystems), the problem can efficiently be solved by lattice reduction algorithms like the LLL algorithm [85]. The Chor-Rivest cryptosystem is an example of cryptosystem which achieves a density close to 1 (for $p = 197$ and $h = 24$, the density is 0.93). Its underlying problem has however the restriction that the subsets must have cardinality equal to h. Refinement of lattice reduction tools with this restriction have been studied by Schnorr and Hörner [144]. They showed that implementations of the Chor-Rivest cryptosystem with parameters $p = 151$ and $h = 16$ could be broken within a few days of computation on a single workstation (in 1995).

So far, the best known attack for secret key recovery is Brickell's attack which works within a complexity of $O(p^{2\sqrt{h}} h^2 \log p)$. It has been published in the final paper by Chor and Rivest [37]. This paper also includes several attempts of attacks when parts of the secret key is disclosed. In Sect. I.5, we briefly review a few of them in order to show what all quantities in the secret key are for.

The Chor-Rivest cryptosystem has the unnatural property that the choice of the finite field $\mathrm{GF}(q)$ must be so that computing the discrete logarithm is easy. A variant has been proposed by Lenstra [84] which overcomes this problem. In this setting, any parameter can be chosen, but the encryption needs multiplications instead of additions. This variant has further been extended by Camion and Chabanne [33].

I.3 Symmetries in the Secret Key

In the Chor-Rivest cryptosystem setting, one has first to choose a random secret key, then to compute the corresponding public key. It relies on the difficulty of finding the secret key from the public key. It shall first be noticed that there are several *equivalent* secret keys, *i.e.* several keys which correspond to the same public key and thus which define the same encryption and decryption functions.

We first notice that if we replace t and g by their pth power (*i.e.* if we apply the Frobenius automorphism in $\mathrm{GF}(q)$), the public key is unchanged because

$$\log_{g^p}(t^p + \alpha_{\pi(i)}) = \frac{1}{p}\log_g((t + \alpha_{\pi(i)})^p) = \log_g(t + \alpha_{\pi(i)}).$$

Second, we can replace (t, α_π) by $(t + u, \alpha_\pi - u)$ for any $u \in \mathrm{GF}(p)$. Finally, we can replace (t, d, α_π) by $(ut, d - \log_g u, u.\alpha_\pi)$ for any $u \in \mathrm{GF}(p)$. Thus we have at least hp^2 equivalent secret keys. The Chor-Rivest problem consists of finding one of it.

Inspired by the symmetry use in the Coppersmith-Stern-Vaudenay attack against birational permutations [40], these properties may suggest that the polynomial $\prod_{i=0}^{h-1}\left(x - t^{p^i}\right)$ of whom all the equivalent t's are the roots plays a crucial role. This is actually the case as shown by the attacks in the following sections.

I.4 Relation to the Permuted Kernel Problem

Throughout this paper, we will use the following property of the Chor-Rivest cryptosystem.

Fact I.1. *For any factor r of h, there exists a generator g_{p^r} of the multiplicative group of the subfield $\mathrm{GF}(p^r)$ of $\mathrm{GF}(q)$ and a polynomial Q with degree h/r and coefficients in $\mathrm{GF}(p^r)$ and such that $-t$ is a root and that for any i we have $Q(\alpha_{\pi(i)}) = {g_{p^r}}^{c_i}$.*

Proof. We let

$$Q(x) = {g_{p^r}}^d \prod_{i=0}^{h/r-1}\left(x + t^{p^{ri}}\right) \tag{I.1}$$

where $g_{p^r} = \prod g^{p^{ri}}$ (g_{p^r} can be considered as the norm of g when considering the extension $\mathrm{GF}(p^r) \subseteq \mathrm{GF}(q)$). We notice that we have $Q(x) \in \mathrm{GF}(p^r)$ for

any $x \in \mathrm{GF}(p^r)$. Since $p^r > \frac{h}{r}$ we obtain that all coefficients are in $\mathrm{GF}(p^r)$. The property $Q(\alpha_{\pi(i)}) = {g_{p^r}}^{c_i}$ is straightforward. □

Since h/r is fairly small, it is unlikely that there exists some other (g_{p^r}, Q) solutions, and g_{p^r} is thus essentially unique. Throughout this paper we will use the notation

$$g_{q'} = g^{\frac{q-1}{q'-1}}.$$

If we consider the Vandermonde matrix

$$M = ({\alpha_i}^j)_{\substack{0 \le i < p \\ 0 \le j \le h/r}}$$

and the vector $V = ({g_{p^r}}^{c_i})_{0 \le i < p}$, we know there exists some vector X such that $M.X = V_{\pi^{-1}}$ where $V_{\pi^{-1}}$ is permuted from V through the permutation π^{-1}. By using the parity check matrix H of the code spanned by M (which is actually a Reed-Solomon code), this can be transformed into a permuted kernel problem $H.V_{\pi^{-1}} = 0$. It can be proved that all entries of H are actually in $\mathrm{GF}(p)$, thus this problem is in fact r simultaneous permuted kernel problems in $\mathrm{GF}(p)$. Actually, we can take $H = (A|I)$ where I is the identity matrix and A is the $(p - h/r - 1) \times (h/r + 1)$-matrix defined by

$$A_{i,j} = -\prod_{\substack{0 \le k < h/r \\ k \ne j}} \frac{\alpha_{i+h/r} - \alpha_k}{\alpha_j - \alpha_k} \quad \begin{pmatrix} 1 \le i < p - h/r \\ 0 \le j \le h/r \end{pmatrix}.$$

If we let V^i denotes the vector of the ith coordinates in vector V, we have

$$\forall i \quad H.V^i_{\pi^{-1}} = 0.$$

Unfortunately, there exists no known efficient algorithms for solving this problem. Since the matrix has a very special form, the author of the present paper believes it is still possible to attack the problem in this direction, which may improve the present attack.

I.5 Partial Key Disclosure Attacks

In this section we show that we can mount an attack when any part of the secret key is disclosed. Several such attacks have already been published in [37]. Some have been improved below and will be used in the following.

Known t Attack. If we guess that $\pi(0) = i$ and $\pi(1) = j$ (because of the symmetry in the secret key, we know that an arbitrary choice of (i, j) will work), we can compute $\log(t + \alpha_i)$ and $\log(t + \alpha_j)$ then solve the equations

$$\begin{aligned} c_0 &= d + \frac{\log(t + \alpha_i)}{\log g} \\ c_1 &= d + \frac{\log(t + \alpha_j)}{\log g} \end{aligned}$$

with unknowns d and $\log g$.[2]

Known g Attack. If we guess that $\pi(0) = i$ and $\pi(1) = j$ (because of the symmetry in the secret key, we know that an arbitrary choice of (i, j) will work), we can compute

$$g^{c_0 - c_1} = \frac{t + \alpha_i}{t + \alpha_j}$$

then solve t.[3]

Known π Attack. We find a linear combination with the form

$$\sum_{i=1}^{p-1} x_i (c_i - c_0) = 0$$

with relatively small integral coefficients x_i's. This can be performed through the LLL algorithm [85]. We can expect that $|x_i| \leq B$ with $B \approx p^{\frac{h}{p-1}}$. Exponentiating this we get some equation

$$\prod_{i \in I} (t + \alpha_{\pi(i)})^{x_i} = \prod_{j \in J} (t + \alpha_{\pi(j)})^{-x_j}$$

with non-negative small powers, which is a polynomial equation with low degree which can be solved efficiently.[4]

Brickell's attack with nothing known consists of finding a similar equation but with a limited number ℓ of $\alpha_{\pi(i)}$ and then exhaustively finding for those $\pi(i)$'s. There is a tradeoff on ℓ: the LLL algorithm may product x_i's smaller than $B = p^{\frac{h}{\ell}}$, the root finding algorithm requires $O(B^2 h \log p)$ GF(p)-operations and the exhaustive search requires $O(p^\ell)$ trials. (For more details and better analysis, see [37].)

[2] Another attack attributed to Goldreich was published in [37].

[3] Another attack was published in Huber [66].

[4] This attack was attributed to Odlyzko and published in [37].

Known g_{p^r} and π Attack. Since we will use this attack several times in the following, we put it here. We can interpolate the $Q(x)$ polynomial of Fact I.1 with $h/r+1$ pairs $(\alpha_{\pi(i)}, {g_{p^r}}^{c_i})$. We thus obtain a h/r-degree polynomial whose roots are conjugates of $-t$. We can thus solve it in order to get t and perform a known t attack.

I.6 Known g_{p^r} Attack

Here we assume we know the g_{p^r} value corresponding to a subfield $\mathrm{GF}(p^r)$ (see Fact I.1).

Let $i_0, \ldots, i_{h/r}$ be $h/r+1$ pairwise distinct indices from 0 to $p-1$. Because of Fact I.1 we can interpolate $Q(x)$ on all $\alpha_{\pi(i_j)}$'s, which leads to the relation

$$ {g_{p^r}}^{c_i} = \sum_{j=0}^{h/r} {g_{p^r}}^{c_{i_j}} \prod_{\substack{0 \le k \le h/r \\ k \neq j}} \frac{\alpha_{\pi(i)} - \alpha_{\pi(i_k)}}{\alpha_{\pi(i_j)} - \alpha_{\pi(i_k)}} \tag{I.2} $$

for $i = 0, \ldots, p-1$. Actually, we can even write this as

$$ {g_{p^r}}^{c_i} - {g_{p^r}}^{c_{i_0}} = \sum_{j=1}^{h/r} ({g_{p^r}}^{c_{i_j}} - {g_{p^r}}^{c_{i_0}}) \prod_{\substack{0 \le k \le h/r \\ k \neq j}} \frac{\alpha_{\pi(i)} - \alpha_{\pi(i_k)}}{\alpha_{\pi(i_j)} - \alpha_{\pi(i_k)}}. \tag{I.3} $$

Because of the symmetry of π in the secret key, we can arbitrarily choose $\pi(i_1)$ and $\pi(i_2)$ (see Sect. I.3).

A straightforward algorithm for finding π consists of exhaustively look for the values of $\pi(i_j)$ for $j = 0, 3, \ldots, h/r$ until Equation (I.2) gives a consistent permutation π. It is illustrated on Fig. I.1. The complexity of this method if roughly $O(hp^{h/r-1})$ computations in $\mathrm{GF}(p)$.

When r is large enough, there is a much better algorithm. Actually, if $h/r \le r$ (*i.e.* $r \ge \sqrt{h}$), the coefficients in Equation (I.2) are the only $\mathrm{GF}(p)$ coefficients which write ${g_{p^r}}^{c_i} - {g_{p^r}}^{c_{i_0}}$ in the basis ${g_{p^r}}^{c_{i_0}} - {g_{p^r}}^{c_{i_0}}, \ldots, {g_{p^r}}^{c_{i_{h/r}}} - {g_{p^r}}^{c_{i_0}}$. Let a_j^i be the coefficient of ${g_{p^r}}^{c_{i_j}} - {g_{p^r}}^{c_{i_0}}$ for ${g_{p^r}}^{c_i} - {g_{p^r}}^{c_{i_0}}$. We have

$$ \frac{a_2^i}{a_1^i} = u \frac{\alpha_{\pi(i)} - \alpha_{\pi(i_1)}}{\alpha_{\pi(i)} - \alpha_{\pi(i_2)}} \tag{I.4} $$

where u is an element of $\mathrm{GF}(p)$ which does not depend on i. Hence, if we randomly choose i_j for $j = 0, \ldots, h/r$, we can write all ${g_{p^r}}^{c_i} - {g_{p^r}}^{c_{i_0}}$'s in the basis $({g_{p^r}}^{c_{i_0}} - {g_{p^r}}^{c_{i_0}}, \ldots, {g_{p^r}}^{c_{i_{h/r}}} - {g_{p^r}}^{c_{i_0}})$. Now if we guess the $\mathrm{GF}(p)$-value of u, we obtain $\pi(i)$ from the above equation. This is a polynomial algorithm in p, h, r for getting π.

Input GF(q) descriptors, α numbering, $c_0, \ldots, c_{p-1}$, $r|h$, g_{p^r}

Output a secret key whose corresponding public key is $c_0, \ldots, c_{p-1}$

1. choose pairwise different $i_0, \ldots, i_{h/r}$ in $\{0, \ldots, p-1\}$
2. choose different $\pi(i_1)$ and $\pi(i_2)$ arbitrarily in $\{0, \ldots, p-1\}$
3. for all the possible values of $\pi(i_0), \pi(i_2), \ldots, \pi(i_{h/r})$ (*i.e.* all values such that $\pi(i_0), \ldots, \pi(i_{h/r})$ are pairwise different and in the set $\{0, \ldots, p-1\}$), we set $S = \{\pi(i_0), \ldots, \pi(i_{h/r})\}$ and do the following
   (a) for all j which is not in S, compute the right-hand term of Equation (I.2) with α_j instead of $\alpha_{\pi(i)}$. If it is equal to ${g_{p^r}}^{c_i}$ such that $\pi(i)$ has not been defined, set $\pi(i) = j$, otherwise continue loop in step 3.
   (b) perform a known g_{p^r} and π attack.

FIGURE I.1 – An $O(p^{\frac{h}{r}-1})$ Known g_{p^r} Attack.

Input GF(q) descriptors, α numbering, $c_0, \ldots, c_{p-1}$, $r|h$, g_{p^r} s.t. $r \geq \sqrt{h}$

Output a secret key whose corresponding public key is $c_0, \ldots, c_{p-1}$

1. choose pairwise different $i_0, \ldots, i_{h/r}$ in $\{0, \ldots, p-1\}$ and precompute the basis transformation matrix for the basis $({g_{p^r}}^{c_{i_0}} - {g_{p^r}}^{c_{i_0}}, \ldots, {g_{p^r}}^{c_{i_{h/r}}} - {g_{p^r}}^{c_{i_0}})$
2. choose different $\pi(i_1)$ and $\pi(i_2)$ arbitrarily in $\{0, \ldots, p-1\}$
3. for all possible u in GF(p), do the following
   (a) for all i, write ${g_{p^r}}^{c_i} - {g_{p^r}}^{c_{i_0}}$ in the basis and get a_0^i and a_1^i. From Equation (I.4) get $\pi(i)$. If it is not consistent with other $\pi(i')$, continue loop in step 3.
   (b) perform a known g_{p^r} and π attack.

FIGURE I.2 – A Polynomial Known g_{p^r} Attack for $r \geq \sqrt{h}$.

Input GF(q) descriptors, α numbering, $c_0, \ldots, c_{p-1}$, $r|h$ s.t. $r \geq \sqrt{h + \frac{1}{4}} + \frac{1}{2}$

Output possible values for g_{p^r}

1. choose pairwise different $i_0, \ldots, i_{h/r}$ in $\{0, \ldots, p-1\}$
2. for any generator g_{p^r} of GF(p^r), do the following
   (a) get the equation of the affine space spanned by $({g_{p^r}}^{c_{i_0}}, \ldots, {g_{p^r}}^{c_{i_{h/r}}})$
   (b) for all other i, check that ${g_{p^r}}^{c_i}$ in the space. If not, continue loop in step 2.
   (c) perform the known g_{p^r} attack of Fig. I.2.

FIGURE I.3 – An $O(p^r)$ Attack for $r \geq \sqrt{h + \frac{1}{4}} + \frac{1}{2}$.

In the rest of the paper, we show how to find g_{p^r} with a choice of r so that these known g_{p^r} attacks can be applied.

I.7 Test for g_{p^r}

Equation (I.3) means that all ${g_{p^r}}^{c_i}$'s actually stand on the same h/r-dimensional affine subspace of GF(p^r) over GF(p). Thus, if we assume that $h/r + 1 \leq r$ (*i.e.* $r \geq \sqrt{h + \frac{1}{4}} + \frac{1}{2}$), this leads to a simple test for g_{p^r}.

Fact I.2. *If there exists a factor r of h such that $r \geq \sqrt{h + \frac{1}{4}} + \frac{1}{2}$ if we let g_{p^r} denotes $g^{1+p^r+\ldots+p^{h-r}}$, then all ${g_{p^r}}^{c_i}$'s stands on the same h/r-dimensional affine space when considering* GF(p^r) *as an r-dimensional* GF(p)*-affine space.*

The existence of such an r can be seen as a bad requirement for this attack, but since the parameters of the Chor-Rivest cryptosystem must make the discrete logarithm easy, we already know that h has many factors, so this hypothesis is likely to be satisfied in practical examples. Actually, h with no such factors are prime and square-prime numbers. The real issue is that r shall not be too large.

Thus there is an algorithm which can check if a candidate for g_{p^r} is good: the algorithm simply check that all ${g_{p^r}}^{c_i}$'s are affine-dependent. The algorithm has an average complexity of $O(h^3/r)$ operations in GF(p). Since there are $\varphi(p^r - 1)/r$ candidates, we can exhaustively search for g_{p^r} within a complexity of $O(h^3 p^r / r^2)$. Since r has to be within the order of $\sqrt{h}$, this attack is better than Brickell's attack provided that such an r exists. The algorithm is depicted on Fig. I.3.

With the parameter $h = 24$, we can take $r = 6$. We have about 2^{41} candidates for g_{p^r} so we can find it within 2^{52} elementary operations, which is feasible with modern computers.

Here we also believe we can still adapt this attack for smaller r values. The next section however gives an alternate shortcut to this issue.

I.8 On the Use of all the c_i's

In his paper [84], Lenstra suspected that disclosing all the c_i's in the public key was a weakness. Actually, this property enables to drastically improve the previous algorithm by using all the factors of h.

We have the following fact.

Fact I.3. *Let $Q(x)$ be a polynomial over $\mathrm{GF}(p^r)$ with degree d and let e be an integer such that $1 \leq e < \frac{p-1}{d}$. We have*

$$\sum_{a \in \mathrm{GF}(p)} Q(a)^e = 0.$$

This comes from the fact that $Q(x)^e$ has a degree less than $p-1$ and that $\sum a^i = 0$ for any $i < p-1$. This proves the following fact.

Fact I.4. *For any $1 \leq e < (p-1)r/h$ we have*

$$\sum_{i=0}^{p-1} {g_{p^r}}^{ec_i} = 0.$$

This provides a much simpler procedure to select all g_{p^r} candidates. Its main advantage is that it works in any subfield. For instance, we can consider $r = 1$ and find the only g_p such that for all $1 \leq e < (p-1)r$ we have $\sum {g_{p^r}}^{ec_i} = 0$. The average complexity of checking one candidate is $O(p)$ $\mathrm{GF}(p)$-computations: it is unlikely that a wrong candidate will not be thrown by the $e = 1$ test. Hence, we can recover g_p within $O(p^2)$ simple computations.

Unfortunately, the g_{p^r} cannot be used efficiently when r is too small. We can still use g_{p^r} in smaller subfields to compute it in large ones. Our goal is to compute g_{p^r} with r large enough. Let us consider the problem of computing g_{p^r} when $r_1, \ldots, r_k$ are factors of r with the knowledge of $g_{p^{r_i}}$. Since we have $g_{p^{r_i}} = g_{p^r}^{1+p^{r_i}+\ldots+p^{r-r_i}}$, we obtain that

$$\log g_{p^r} = \frac{\log g_{p^{r_i}}}{1 + p^{r_i} + \ldots + p^{r-r_i}} \pmod{p^{r_i} - 1}. \tag{I.5}$$

Input GF(q) descriptors, α numbering, $c_0, \ldots, c_{p-1}$, $r_i | r | h$ and $g_{p^{r_i}}$, $i = 1, \ldots, k$

Output set of possible g_{p^r} values

1. solve the Equation System (I.5) for $i = 1, \ldots, k$ and obtain that $g_{p^r} = \beta.\gamma^x$ for unknown x
2. for $x = 0, \ldots, (p^r - 1)/\mathrm{lcm}\{p^{r_i} - 1; i = 1, \ldots, k\} - 1$ do the following
   (a) compute $\sum \beta^{ec_i} \gamma^{ec_i x}$ for $e = 1, \ldots (p-1)r/h - 1$ and if one sum is non-zero continue loop on step 2.
   (b) output $g_{p^r} = \beta.\gamma^x$

FIGURE I.4 – Getting g_{p^r} from the $g_{p^{r_i}}$.

The knowledge of all $g_{p^{r_i}}$'s thus gives the knowledge of $\log g_{p^r}$ modulo

$$\ell = \mathrm{lcm}\{p^{r_1} - 1, p^{r_2} - 1, \ldots, p^{r_k} - 1\}.$$

Thus we need only $(p^r - 1)/\ell$ trials to recover g_{p^r}. The algorithm is illustrated on Fig. I.4. It is easy to see that each loop controlled in step 2 requires on average $O(pr^2)$ operations in GF(p).

Thus we can define an algorithm for dedicated h's by a graph.

Definition I.5. *Let G be a rooted labeled direct acyclic graph in which the root is labeled by a finite field* GF(p^r) *and such that whenever there is a $u \to v$ edge in G then the label $L(u)$ of u is a subfield of the label $L(v)$ of v and an extension of* GF(p). *We call G a "p-factoring DAG for* GF(p^r)*".*

To G and an integer p we associate the quantity

$$C(G) = \sum_v \frac{\#L(v) - 1}{\mathrm{lcm}\{\#L(w) - 1; v \leftarrow w\}}.$$

(By convention, lcm of an empty set is 1.) We can define an algorithm for computing g_{p^r} with complexity $O(pr^2 C(G))$. Thus, we can break the Chor-Rivest cryptosystem with parameter h which is neither prime nor a square prime within a complexity essentially

$$O\left(\min_{\substack{r|h \\ r \geq \sqrt{h}}} \min_{\substack{G \text{ is a } p-\text{factoring} \\ \text{DAG for GF}(p^r)}} pr^2 C(G)\right).$$

The corresponding algorithm is illustrated on Fig. I.5.

Input GF(p^h) descriptors, α numbering, $c_0, \ldots, c_{p-1}$,

Output a possible secret key

1. for the smallest factor r of h such that $r \geq \sqrt{h + \frac{1}{4}} + \frac{1}{2}$, find the p-factoring DAG with minimal $C(G)$
2. for any u in G such that for all $u \leftarrow u_i$, u_i has been visited, visit u by doing the following
   (a) perform the algorithm of Fig. I.4 with GF(p^r) $= L(u)$ and GF(p^{r_i}) $= L(u_i)$ and obtain g_{p^r}
3. perform the known g_{p^r} attack of Fig. I.2

FIGURE I.5 – An Efficient Attack Dedicated for h.

Example I.6 ($h = 25$). We can solve the $h = 25$ case with a trivial G p-factoring DAG for GF(p^5) which consists of two vertices labeled with GF(p) and GF(p^5). From g_{p^5} we can then apply the algorithm of Fig. I.2. We have

$$C(G) = \frac{p^5 - 1}{p - 1} + p - 1 \approx p^4$$

so the corresponding complexity is $O(p^5)$.

Example I.7 ($h = 24$). Here is another dedicated attack for $h = 24$. We can choose $r = 6$ for which we have $h/r + 1 \leq r$. Recovering g_{p^6} requires firstly, $O(p)$ trials to get g_p, secondly, $O(p)$ trials to get g_{p^2} with g_p, thirdly, $O(p^2)$ trials to get g_{p^3} with g_p, finally, $O(p^2)$ trials to get g_{p^6} with g_{p^2} and g_{p^3}. The maximum number of trials is thus $O(p^2)$. Hence the complexity is $O(p^3)$ multiplications in GF(p^6). Actually, this attack corresponds to the p-factoring DAG for GF(p^6) depicted on Fig. I.6. For this DAG we have

$$C(G) = \frac{p^6 - 1}{\mathrm{lcm}(p^2 - 1, p^3 - 1)} + \frac{p^3 - 1}{p - 1} + \frac{p^2 - 1}{p - 1} + p - 1$$

thus $C(G) = 78014$ for $p = 197$. We thus need about 2^{29} operations in $GF(197)$ to break the Chor-Rivest cryptosystem in GF(197^{24}).

I.9 Generalization

In this section we generalize our attack in order to cover the GF(256^{25}) case *i.e.* when p is a power-prime: there is no reason why to restrict our

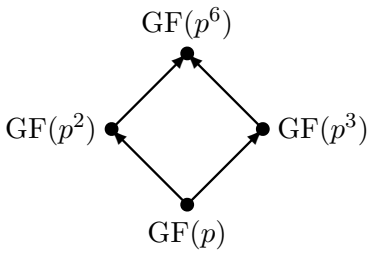


FIGURE I.6 – A Factoring DAG for GF(p^6).

attacks to finite fields which are extensions of GF(p) since we have many other subfields. For this we need to adapt the algorithm of Fig. I.5 with generalized factoring DAGs, *i.e.* when the labels are not extensions of GF(p). We first state generalized version of Fact I.1.

Fact I.8. *Let* GF(q') *be a subfield of* GF(q) *i.e.* $q = q'^s$. *We let*

$$Q(x) = N(g^d(x+t)) \bmod (x^p - x)$$

where $N(y) = y^{\frac{q-1}{q'-1}}$. $Q(x)$ *is a polynomial such that* $Q(\alpha_{\pi(i)}) = N(g)^{c_i}$. *In addition, if we have* $\gcd(s,h) < p_0$ *where* $p_0 = q^{\frac{1}{\mathrm{lcm}(s,h)}}$ *then the degree of* $Q(x)$ *is* $\gcd(s,h)\frac{p-1}{p_0-1}$.

Proof. $Q(\alpha_{\pi(i)}) = N(g)^{c_i}$ is obvious since $\alpha_{\pi(i)}$ is a root of $x^p - x$. The useful part of this fact is the distance between the degree of $Q(x)$ and p.

We have

$$Q(x) \equiv N(g).N(x+t) \equiv N(g) \prod_{i=0}^{s-1} \left(x^{q'^i} + t^{q'^i}\right) \pmod{(x^p - x)}.$$

We notice that

$$x^i \bmod (x^p - x) = x^{(i-1) \bmod (p-1)+1}$$

thus if we let

$$d = \sum_{i=0}^{s-1} \left(\left(q'^i - 1\right) \bmod (p-1) + 1\right)$$

the degree of $Q(x)$ is d provided that $d < p$. Let $p_0 = q^{\frac{1}{\mathrm{lcm}(s,h)}}$ and $p = p_0^g$. We have

$$d = \frac{s}{g} \sum_{i=0}^{g-1} \left(\left(p_0^i - 1\right) \bmod (p_0^g - 1) + 1\right) = \frac{s}{g} \sum_{i=0}^{g-1} p_0^i = \frac{s}{g} \frac{p-1}{p_0 - 1}.$$

We further notice that $\frac{s}{g} = \gcd(s,h)$ and that $d < p$. □

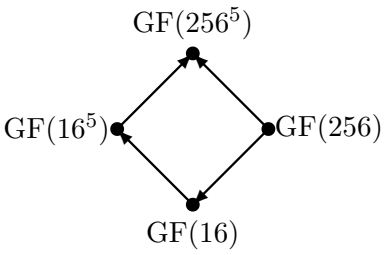


FIGURE I.7 – A Generalized Factoring DAG for GF(256^5).

As a consequence we obtain a generalized form of Fact I.4.

Fact I.9. *Let* $q = p^h = q'^s$ *and* $p_0 = q^{\frac{1}{\mathrm{lcm}(s,h)}}$ *be such that* $\gcd(s,h) < p_0 - 1$. *We have*

$$\sum_{i=0}^{p-1} {g_{q'}}^{ec_i} = 0$$

for any $1 \leq e < \frac{p_0-1}{\gcd(s,h)}$.

We can thus generalize the attack of Fig. I.5 whenever each GF($q^{1/s}$) label fulfill the assumption $\gcd(s,h) < p_0 - 1$ where $p_0 = q^{\frac{1}{\mathrm{lcm}(s,h)}}$.

Example I.10 ($q = 256^{25}$). The GF(16) field does not fulfill the assumption. However, the GF(256), GF(16^5) and GF(256^5) fields do. We can thus start the attack with the field GF(256) and then obtain g_{16} from g_{16^2} as illustrated by the (generalized) factoring DAG of GF(256^5) illustrated on Fig. I.7. We have

$$C(G) = \frac{256^5 - 1}{\mathrm{lcm}(255, 16^5 - 1)} + \frac{16^5 - 1}{15} + \frac{15}{255} + 255 = 131841 + \frac{1}{17}$$

thus we need about 2^{29} GF(16)-operations to break the Chor-Rivest cryptosystem in GF(256^{25}).

We believe there is no need for formalizing further generalizations in the Chor-Rivest cryptosystem context. We believe that the more we have some subfield choices of GF(q), the lower is the complexity of the best attack.

I.10 Conclusion

We have described general attack when the parameter h has a small factor r greater than $\sqrt{h + \frac{1}{4}} + \frac{1}{2}$ which has a complexity $O(h^3 p^r / r^2)$. We also have

solved one of Lenstra's conjectures arguing that keeping all the c_i coefficients in the public key is a weakness by exhibiting a shortcut algorithm in the previous attack.

The attack has been successfully implemented on an old laptop with the suggested parameters GF(p^{24}) by using hand-made (inefficient) arithmetic libraries. Recovering the secret key from the public key takes about 15 minutes. But computing the public key from the secret key takes much longer...

We also generalized our attack in order to break the GF(256^{25}) proposal. In Appendix, we even suggest an improvement of the presented attacks when h does not have a small factor r greater than $\sqrt{h + \frac{1}{4}} + \frac{1}{2}$.

In order to repair the Chor-Rivest cryptosystem, we believe that

- we must choose a finite field GF(p^h) where p and h are both prime;
- we must not put all the c_is in the public key.

It is then not clear how to choose the parameters in order to make the discrete logarithm problem easy, and to achieve a good knapsack density in order to thwart the Schnorr-Hörner attack.

One solution is to use Lenstra's Powerline cryptosystem, or even its recent generalization: the Fractional Powerline System (see Camion-Chabanne [33]). We however have to fulfill the two requirements above. The security in this setting is still open, but we suspect that the simultaneous permuted kernel characterization of the underlying problem may lead to a more general attack on this cryptosystem with any parameters. We highly encourage further work in this direction.

Acknowledgment

The author thanks Andrew Odlyzko for many helpful discussions and AT&T for inviting to perform those researches.

I.A Extension of Algorithm of Fig. I.2

Equation (I.4) is a simple way to solve the problem when $r \geq \sqrt{h}$. We still believe we can adapt the above attack for any value of r by more tricky algebraic computations.

Actually, let us consider a value r such that $\frac{h}{r} \geq r$ and $\ell = \frac{h}{r} - r$. Let e_i denotes $g_{p^r}{}^{c_{i_j}} - g_{p^r}{}^{c_{i_0}}$ for $i = 1, \ldots, h/r$. There may exist some $\sum_j u_{k,j} e_j = 0$

equations, namely ℓ of it. Hence if we write ${g_{p^r}}^{c_i} - {g_{p^r}}^{c_{i_0}} = \sum_j a_j^i e_j$, there may exist some x_k^i coefficients such that

$$a_j^i - \sum_k x_k^i u_{k,j} = \prod_{\substack{0 \le k \le h/r \\ k \ne j}} \frac{\alpha_{\pi(i)} - \alpha_{\pi(i_k)}}{\alpha_{\pi(i_j)} - \alpha_{\pi(i_k)}}$$

for $j = 1, \ldots, h/r$. When considering a set of n values of i, we have nh/r algebraic equations with $n(\ell+1) - 1 + h/r$ unknowns x_k^i, $\alpha_{\pi(i_j)}$, $\alpha_{\pi(i)}$. Thus if $r > 1$ we can take n large enough as long as $p(r-1) + 1 \ge h/r$. We thus believe further algebraic tricks may leads to the solution for any $r > 1$ as long as $p + 1 \ge h/2$.

Annexe J

Security of the Birational Permutation Signature Schemes

[*Cet article de* Don Coppersmith, Jacques Stern *et* Serge Vaudenay *a été publié dans le* Journal of Cryptology *en 1997 [40] après avoir été présenté au colloque Crypto 93 [39].*]

Abstract. In recent years, researchers have invested a lot of effort in trying to design suitable alternatives to the RSA signature scheme, with lower computational requirements. The idea of using polynomial equations of low degree in several unknowns, with some hidden trapdoor, has been particularly attractive. One of the most noticeable attempt to push this idea forward is the Ong-Schnorr-Shamir signature scheme, which has been broken by Pollard and Schnorr. At Crypto'93, Shamir proposed a family of cryptographic signature schemes based on a new method. His design made subtle use of birational permutations over the set of k-tuples of integers modulo a large number N of unknown factorization. However, the schemes presented in Shamir's paper are weak. In the present paper, we describe several attacks which can be applied to schemes in this general family.

Introduction

The celebrated RSA cryptosystem can be viewed as a permutation computed in both directions as a polynomial over the ring $\mathbf{Z}_N$, where N is a (large) integer with secret factorization. In search for suitable alternatives to the RSA signature scheme, with lower computational requirements, several cryptographers have suggested to use polynomials of low degree in several variables.

In the context of signature, such polynomials were natural candidates for the design of very efficient schemes, both for signature generation and signature verification.

The first cryptographic protocol based on this principle is the Ong-Schnorr-Shamir signature scheme [117]. It has been broken by Pollard and Schnorr [126]. At Crypto'93, Shamir proposed a family of cryptographic signature schemes based on a new method. His design made subtle use of birational permutations of the integers modulo N. Shamir actually introduced several techniques: the first technique uses as a trap-door a family of quadratic forms built in a very specific way and which he calls *sequentially linearized*. It is a kind of generalization of the Ong-Schnorr-Shamir scheme, with more unknowns and more equations to be solved for signature generation. Another technique uses the notion of an algebraic basis for the quadratic forms: this is a set of quadratic forms from which any other one can be computed by using only rational operations. This technique can be further divided according to the algebraic basis chosen and Shamir's paper includes two proposals, one using a symmetric basis and the other an asymmetric one. Of course, there is nothing specific to quadratic forms in Shamir's approach: it only turns out that use of cubic or quartic polynomials makes key management cumbersome and loses the computational advantages shown by the scheme.

In the present paper, we show that the schemes presented in Shamir's paper are weak, by exhibiting several attacks which can be applied to schemes in the general family. These results have been announced in [39], where we deal with the trapdoor based on sequentially linearized equations and with the symmetric basis proposal. Since then, another attack has appeared in [159], which takes care of the asymmetric basis.

It is worth mentioning that another public key system scheme based on quadratic forms has been proposed by Matsumoto and Imai [95]. This scheme is based on completely different ideas and uses (small) fields of characteristic 2. Let us add that the Matsumoto-Imai scheme has recently been broken by Patarin [120]. Thus, there seems to be some kind of intrinsic difficulty that prevents hiding trap-doors into families of quadratic forms.

We close this introduction by thanking Adi Shamir both for sending us his Crypto'93 paper at an early stage and for many discussions on the subject of this paper.

J.1 The Methodology of the Attacks.

J.1.1 The Overall Strategy.

Basically, Shamir's idea is to start from a family of quadratic forms with some "visible" algebraic structure (e.g. low rank) and to hide the underlying structure by performing the following operations

1. linear change of coordinates
2. linear combinations of the resulting forms

We are thus faced to the problem of trying to recapture some of the hidden structure, from the public key only. This public key consists of several quadratic forms and we note that, as a consequence of step 2 above, some linear combinations of the public forms may retain a part of the original algebraic structure. Unfortunately, we can only handle these objects indirectly, through the use of indeterminate coefficients, say $\delta, \epsilon, \ldots$. At this point we note that many of the properties used in the design proposed by Shamir can be expressed by the vanishing of polynomials in $\delta, \epsilon, \ldots$. We quote several examples:

- the fact that a quadratic form has not full rank is expressed by the vanishing of its determinant
- the fact that a quadratic form has rank 2 is expressed by the vanishing of all 3×3 determinants
- the fact that a vector u belongs to the vector space spanned by the rows of the matrix M of a given quadratic form of rank k is expressed by the vanishing of all $(k+1) \times (k+1)$ determinants of the matrix M' obtained by appending u as an extra row to M. These determinants are polynomials in $\delta, \epsilon, \ldots$ and in the coordinates of u.

J.1.2 Galois Theory and Ideal Calculations.

We are thus led to a set of polynomial equations in $\delta, \epsilon, \ldots$. Such a set of equations generates an ideal in the ring of polynomial with several unknowns: in other words, if $P_1, \ldots, P_m$ are m polynomials with unknowns $\delta, \epsilon, \ldots$, the equations $P_i = 0$ define an algebraic curve associated to the ideal of all polynomials which can be written

$$P_1 Q_1 + \ldots + P_m Q_m$$

where $Q_1, \ldots, Q_m$ are arbitrary polynomials.

At this point, we have to return to the underlying structure. In case there is a lack of symmetry, as in Shamir's first scheme, we can try to solve for one of the unknowns: this simply means that the ideal should contain a polynomial of degree one with a single variable. In other cases, we observe a strong symmetry: for example, in the scheme based on the symmetric basis, we isolate a sequence of integers modulo N, say $\delta_1, \ldots, \delta_k$, coming from the hidden structure, which act as (say) first coordinates of points $m_1, \ldots, m_k$ of the curve which cannot be distinguished from each other. In such a case, it is hopeless to try to solve for the first coordinate δ. On the other hand, we expect to find in the ideal a polynomial of degree k in the single variable δ, $F(\delta)$, which we can treat symbolically and of which the values $\delta_1, \ldots, \delta_k$ are the unknown roots. This is a context close to Galois theory. Still, we do not really offer proofs of the various statements we make relying on Galois-like arguments. Although it might be possible to write up proofs in some cases, we feel that the technicalities would distract from the issues at hand. In place, we remain at an informal level and implicitly assume a large degree of "genericity". We think that this is perfectly acceptable in a paper concerned with cryptanalysis: furthermore, our attack has been implemented using a computer algebra package, and this is a kind of experimental verification of the correctness of our statements.

We now turn to ideal calculations. As explained above, we need to disclose members of the ideal with prescribed degrees for the various unknowns. For this, we can use Gröbner basis algorithms (see [42]), which output another family of polynomials $P'_1, \ldots, P'_r$ spanning the same ideal and which is *reduced* in a suitable sense. The drawback of this algorithm is its high complexity. In case we are trying to eliminate all unknowns except one, we can repeatedly form resultants of two polynomials with respect to a given unknown. We can also apply the Euclidean algorithm to compute the g.c.d. of two polynomials with respect to to a given unknown. This decreases the degree of an equation.

Finally, it is also possible to use a simpler *ad hoc* version of the Gröbner basis algorithm which is a kind of generalized Gaussian elimination. For instance, if M is a monomial in P_1 which does not divide any other monomial of P_1, then every multiple monomial of M can be eliminated in the other polynomials by replacing P_i by $P'_i = P_i - Q_i P_1$ for a suitable polynomial Q_i. Then, one can continue the reduction with $P'_2, \ldots, P'_m$. In most cases, if m is large enough and if there is a hidden trap-door in the set of equations, this reduction is likely to end rather quickly with very simple equations of the expected form. In the rest of the paper, we will not comment further on ideal calculations and will thus treat them as a kind of "programming technology", which we actually used in our experiments.

J.1.3 Working mod N versus Working mod p.

Our analysis basically treats the ring of integers mod N as a field. Actually, N is composite and we assume for simplicity that it has only two prime factors p and q. Our calculations make sense mod p since we are actually working in a field but some justification is needed to go from calculations mod p to calculations mod N. In section J.2, we will only use tools from linear algebra such as Gaussian elimination or determinants. Thus all computations go through regardless of the fact that N is composite. The situation is a bit more subtle in section J.3, where Galois theory comes into the picture. For instance, assume that we have discovered a polynomial $F(\delta)$ of degree k from a sequence of integers modulo N, say $\delta_1, \ldots, \delta_k$, coming from the hidden structure, as explained in section J.1.2. Such a polynomial has k solutions mod p but k^2 solutions mod N, each obtained by mixing some solution mod p with some solution mod q. But if we consider only the image, mod p, of our calculations mod N, things are all right. As will be shown, our cryptanalysis provides a way to forge signatures by performing calculations which treat δ (and possibly other variables) symbolically. Galois-like arguments show that the result has the expected symmetry and thus, is expressible in terms of the coefficients of F and in terms of the coefficients of the public key. These calculations are valid mod p. They are also valid mod q, and the Chinese remainder theorem suffices to make them valid mod N. This is in spite of the fact that a solution δ of F mod N might well mix different solutions δ_i mod p and δ_j mod q. Since we never explicitly solve for δ, but only work with it symbolically and use the fact that $F(\delta) = 0 \bmod N$, we never are in danger of factoring N.

J.2 The First Scheme

The first family of Shamir's signature schemes is based on sequentially linearized equations. The public information consists of a large integer N of unknown factorization (even the legitimate users need not know its factorization), and the coefficients of $k-1$ quadratic forms $f_2, \ldots, f_k$ in k variables $x_1, \ldots, x_k$ each. Each of these quadratic forms can be written as

$$f_i = \sum_{j,\ell} \alpha_{ij\ell} x_j x_\ell \tag{J.1}$$

where i ranges from 2 to k and the matrix $\alpha_{ij\ell}$ is symmetric i.e. $\alpha_{ij\ell} = \alpha_{i\ell j}$.

The secret information is a pair of linear transformations. One linear transformation B relates the quadratic forms $f_2, \ldots, f_k$ to another sequence of quadratic forms $g_2, \ldots, g_k$. The second linear transformation A is a change

of coordinates that relates the variables $(x_1, \ldots, x_k)$ to a set of "original" variables $(y_1, \ldots, y_k)$. Denoting by Y the column vector of the original variables and by X the column vector of the new variables, we can simply write $Y = AX$.

Of course, the coefficients of A and B are known only to the legitimate user. The trap-door requirements are twofold: when expressed in terms of the original variables $y_1, \ldots, y_k$, the quadratic form g_2 is computed as:

$$g_2 = y_1 y_2 \tag{J.2}$$

and the subsequent g_i's, $3 \leq i \leq k$ are *sequentially linearized*, i.e. can be written

$$g_i(y_1, \ldots, y_k) = \ell_i(y_1, \ldots, y_{i-1}) \times y_i + q_i(y_1, \ldots, y_{i-1}) \tag{J.3}$$

where ℓ_i is a linear function of its inputs and q_i is a quadratic form.

To sign a message M, one hashes M to a $k-1$-tuple $(f_2, \ldots, f_k)$ of integers modulo N, then finds a sequence $(x_1, \ldots, x_k)$ of integers modulo N satisfying (J.1). This is easy from the trap-door.

It is straightforward that the particular case $k = 2$ is equivalent to the Ong-Schnorr-Shamir scheme [117]. The Pollard-Schnorr algorithm [126] enables to forge a valid signature of any message. In the following, we show how to break the other cases reducing them to the Ong-Schnorr-Shamir scheme too.

We let M_i, $2 \leq i \leq k$ denote the $k \times k$ symmetric matrix of the quadratic form g_i. The kernel K_i of g_i is the kernel of the linear mapping whose matrix is M_i. It consists of vectors which are orthogonal to all vectors with respect to g_i. The rank of the quadratic form g_i is the rank of M_i. It is the codimension of K_i as well as the unique integer r such that g_i can be written as a linear combination of squares of r independent linear functionals. (For more details, see [82] for instance.) Actually, all this is not completely accurate as N is not a prime number and therefore $\mathbf{Z}_N$ is not a field. This question has been addressed in section J.1.3 and we now ignore the problem.

An easy computation shows that K_i is the subspace defined in terms of the original variables by the equations

$$y_1 = \ldots = y_i = 0 \tag{J.4}$$

From this, it follows that
i) K_i is decreasing;
ii) the dimension of K_i is $k - i$;

iii) any element of K_{i-1} not in K_i is an isotropic element wrt g_i, which means that the value of g_i is zero at this element.

We will construct a basis b_i of the k-dimensional space, such that the family $b_{i+1}, \ldots, b_k$ spans K_i for $i = 2, \ldots, k-1$. The main problem we face is the fact that the g_i's and therefore the K_i's are unknown. Instead, we know the f_i's. We concentrate on the (unknown) coefficient δ_i of g_k in the expression of f_i, i.e. we write

$$f_i = \delta_i g_k + \sum_{j=2}^{k-1} \beta_{ij} g_j \tag{J.5}$$

As coefficients have been chosen randomly, we may assume that δ_k is not zero. Let $i < k$. Consider the quadratic form (in all x_is) $Q_i(\lambda) = f_i - \lambda f_k$. When $\lambda = \delta_i/\delta_k$, this form has a non-trivial kernel and therefore δ_i/δ_k is a root of the polynomial $P_i(\lambda) = \det(Q_i(\lambda))$. This is not enough to recover the correct value of λ. Computing the matrix of $Q_i(\lambda)$ for $\lambda_i = \delta_i/\delta_k$ in the basis corresponding to the original coordinates $y_1, \ldots, y_k$ yields the following

$$\begin{pmatrix} \boxed{C} & \boxed{0} \\ \boxed{0} & 0 \end{pmatrix}$$

In the same basis, the matrix of $Q_i(\lambda)$ for any λ, can be written as

$$\begin{pmatrix} \boxed{C_\lambda} & \boxed{U_\lambda} \\ \boxed{(U_\lambda)^t} & 0 \end{pmatrix}$$

We observe that U_λ is affine in λ and vanishes at λ_i so that the determinant of the matrix is divisible by $(\lambda - \lambda_i)^2$. Since determinants can be computed up to a multiplicative constant in any basis, it follows that $(\lambda - \lambda_i)^2$ factors out in $P_i(\lambda)$. Thus the correct value of λ_i can be found by observing that it is a double root of the polynomial equation $P_i(\lambda) = 0$. We

now make use of the informal genericity principle explained in section J.1.2, which means that we ignore "exceptional" situations. As a consequence, we claim that the double root is disclosed by simply taking the g.c.d. in $\mathbf{Z}_N$ of P_i and P_i' with respect to λ. We find a linear equation in λ, from which we easily compute λ_i.

Once all coefficients λ_i have been recovered, we set for $i = 2, \ldots, k-1$

$$\tilde{f}_i = f_i - \lambda_i f_k \tag{J.6}$$

and $\tilde{f}_k = f_k$. We note that all quadratic forms $\tilde{f}_i$ have kernel K_{k-1}. This allows to pick a non-zero vector b_k in K_{k-1}. The construction can then go on inductively in the quotient space of the k-dimensional space by the vector spanned by $\{b_k\}$ with $\tilde{f}_2, \ldots, \tilde{f}_{k-1}$ in place of $f_2, \ldots, f_k$.

At the end of the recursive construction, we obtain a sequence b_i, $3 \leq i \leq k$ such that $b_{i+1}, \ldots, b_k$ spans K_i for $i = 2, \ldots, k-1$ and a sequence of quadratic forms $\tilde{f}_2, \ldots, \tilde{f}_k$ such that
i) $\tilde{f}_i$ has kernel K_i;
ii) b_i is an isotropic element wrt $\tilde{f}_i$.
Choosing b_1, b_2 at random, we get another set of coordinates $z_1, \ldots, z_k$ defined by $X = (b_1 \ldots b_n)Z$ such that
i) $\tilde{f}_2$ is a quadratic form in the coordinates z_1, z_2
ii) $\tilde{f}_3, \ldots, \tilde{f}_k$ is sequentially linearized

The rest is easy. From a sequence of prescribed values for $f_2, \ldots, f_k$, we can compute the corresponding values of $\tilde{f}_2, \ldots, \tilde{f}_k$. Next, we can find values of $\{z_1, z_2\}$ achieving a given value of $\tilde{f}_2 \bmod N$ in exactly the same way as the Pollard solution of the Ong-Schnorr-Shamir scheme [117]. Then, values for $z_3, \ldots, z_k$ achieving given values of $\tilde{f}_3, \ldots, \tilde{f}_k$ are found by successively solving $k-2$ linear equations. Finally, the values of $z_1, \ldots, z_k$ can be translated into values of $x_1, \ldots, x_k$.

Example. In Shamir's paper [149], an example is given with $N = 101$ (the fact that 101 is prime is unfortunate but actually irrelevant).

$$v_2 = 78x_1^2 + 37x_2^2 + 6x_3^2 + 54x_1x_2 + 19x_1x_3 + 11x_2x_3 \pmod{101}$$

$$v_3 = 84x_1^2 + 71x_2^2 + 48x_3^2 + 44x_1x_2 + 33x_1x_3 + 83x_2x_3 \pmod{101}.$$

Matrices of f_2, f_3 are as follows

$$\begin{pmatrix} 78 & 27 & 60 \\ 27 & 37 & 56 \\ 60 & 56 & 6 \end{pmatrix} \qquad \begin{pmatrix} 84 & 22 & 67 \\ 22 & 71 & 92 \\ 67 & 92 & 48 \end{pmatrix}$$

We get:

$$P(\lambda) = \det(f_2 - \lambda f_3) = 34(\lambda^3 + 75\lambda^2 + 55\lambda + 71) \tag{J.7}$$

$$P'(\lambda) = \lambda^2 + 50\lambda + 52 \tag{J.8}$$

$$\gcd(P, P') = \lambda - 63 \tag{J.9}$$

We let

$$\tilde{f}_2 = f_2 - 63 f_3 \quad ; \quad \tilde{f}_3 = f_3 \tag{J.10}$$

The kernel of $\tilde{f}_2$ is spanned by vector $b_3 = (31, 12, 1)^t$. We pick $b_2 = (0, 1, 0)^t$ and $b_1 = (1, 31, 0)^t$. We get, in the corresponding coordinates z_1, z_2, z_3:

$$\tilde{f}_2 = 26z_1^2 + 8z_2^2 \quad ; \quad \tilde{f}_3 = z_3(26z_1 + 20z_2) + 90z_1^2 + 2z_1z_2 + 71z_2^2 \tag{J.11}$$

Then, for any tuple (f_2, f_3) of integers, we compute $(\tilde{f}_2, \tilde{f}_3)$ using (J.10), we solve $\tilde{f}_2 = 26z_1^2 + 8z_2^2$ by Pollard-Schnorr's algorithm and compute z_3 such that equations (J.11) hold. This forges a signature.

J.3 The Second Scheme

We now treat Shamir's [149] second scheme. Throughout, we will pretend we are working in $\mathbf{Z}_p$ rather than $\mathbf{Z}_N$. This has been explained in section J.1.3.

We briefly review the scheme. Shamir begins with k variables $y_1, y_2, \ldots, y_k$, with k odd. These are subjected to a secret linear change of variables which gives $u_i = \sum_j a_{ij} y_j, i = 1, 2, \ldots, k$, with the matrix $A = (a_{ij})$ secret. The products $u_i u_{i+1}$, including $u_k u_1$, are subjected to a second secret linear transformation $B = (b_{ij})$, so that $v_i = \sum_j b_{ij} u_j u_{j+1}, i = 1, 2, \ldots, k - s$. The public key is the set of coefficients $(c_{ij\ell})$ expressing v_i in terms of pairwise products $y_j y_\ell$, for $1 \leq i \leq k - s$,

$$v_i = \sum_{j,\ell} c_{ij\ell} y_j y_\ell, 1 \leq i \leq k - s, c_{ij\ell} = c_{i\ell j} \tag{J.12}$$

In the above, $s \geq 1$ is a parameter and, for the sake of simplicity, we treat first the case $s = 1$. Thus, i is ranging to $k - 1$, meaning that we have discarded $s = 1$ of the v_i. A valid signature of a $(k-1)$-tuple of integers $(v_1, \ldots, v_{k-1})$ is a set of values of $y_1 y_2, \ldots, y_k y_1$ such that (J.12) holds. Signature generation for the legitimate user is based on the fact that $y_1 y_2, \ldots, y_k y_1$ form an algebraic

basis for the ideal generated by quadratic forms: for example, if $k = 3$, y_1^2 is recovered by the formula

$$y_1^2 = \frac{(y_1 y_2)(y_3 y_1)}{y_2 y_3}$$

See [149] for more details.

The first step in our solution is as follows: linear combinations of the v_i are linear combinations of the $u_i u_{i+1}$, but they form only a subspace of dimension $k-1$. Some linear combinations of the v_i,

$$v_1 + \delta v_2 + \sum_{3 \le j \le k-1} \epsilon_j v_j \tag{J.13}$$

will be quadratic forms in the y_i of rank 2. A computation shows that the only linear combinations of the products $u_i u_{i+1}$ of rank 2 are of the form

$$\alpha_i u_{i-1} u_i + \beta_i u_i u_{i+1} = u_i(\alpha_i u_{i-1} + \beta_i u_{i+1}) \tag{J.14}$$

for any values of α_i, β_i, i. Because the v_j span a subspace of codimension 1, and because we are further restricting to one lower dimension by the choice of the multiplier 1 for v_1 in the linear combination, we find that for each i there will be one pair (α_i, β_i) and one set of coefficients $(\delta_i, \epsilon_{ij})$ such that

$$\alpha_i u_{i-1} u_i + \beta_i u_i u_{i+1} = u_i(\alpha_i u_{i-1} + \beta_i u_{i+1}) = v_1 + \delta_i v_2 + \sum_{3 \le j \le k-1} \epsilon_{ij} v_j \tag{J.15}$$

We now omit the i indices for the sake of clarity. The condition of being rank 2 is an algebraic condition: setting

$$v_1 + \delta v_2 + \sum_{3 \le j \le k-1} \epsilon_j v_j = \sum_{j\ell} \tau_{j\ell} y_j y_\ell \tag{J.16}$$

with $\tau_{j\ell} = \tau_{\ell j}$, we find that each 3×3 submatrix of the matrix $(\tau_{j\ell})$ has vanishing determinant. Each of these determinants is a polynomial equation in δ, ϵ_j. Use resultants and Gaussian elimination to eliminate ϵ_j from this family of polynomial equations (in the ring $\mathbf{Z}_N$) and find a single polynomial F of degree k satisfied by δ. We also find ϵ_j as polynomials in δ, by returning to the original equations and eliminating the variables $\epsilon_i, i \neq j$.

Thus each solution δ to $F(\delta) = 0$ gives rise to a linear combination of v_j which is of rank 2. The root δ corresponds to that index i for which

$$v_1 + \delta v_2 + \sum_{3 \le j \le k-1} \epsilon_j v_j = u_i(\alpha_i u_{i-1} + \beta_i u_{i+1}) \tag{J.17}$$

We will indicate this correspondence by writing $\delta = \delta_i$.

For each solution $\delta = \delta_i$, the rows of the resulting matrix (τ_{ij}) span a subspace $Y(\delta_i) = Y_i$ of $\mathbf{Z}_p{}^k$ of rank 2; namely, Y_i is spanned by u_i and $\alpha_i u_{i-1} + \beta_i u_{i+1}$. This observation is rather straightforward if the quadratic form is expressed in the basis corresponding to the u_i variables. Going to the y_i coordinates involves a right multiplication by A and a left multiplication by its transpose A^t. The first operation replaces row vectors by their expressions in terms of the "new" variables y_i and the latter does not affect the vector space spanned by the rows. This is enough to conclude.

Observe that u_i, u_{i+2}, and $(\alpha_{i+1}u_i + \beta_{i+1}u_{i+2})$ are linearly related, as are u_i, u_{i-2}, and $(\alpha_{i-1}u_{i-2} + \beta_{i-1}u_i)$. So

$$u_i \in Y_i \cap (Y_{i+1} + Y_{i+2}) \cap (Y_{i-1} + Y_{i-2}) \tag{J.18}$$

This is an algebraic relation among δ_{i-2}, δ_{i-1}, δ_i, δ_{i+1}, and δ_{i+2}. More accurately, for $k > 5$, $(i+1, i+2)$ and $(i-1, i-2)$ are the only instances of pairs (a, b) (c, d) consisting of four different indices, all distinct from i, such that

$$Y_i \cap (Y_a + Y_b) \cap (Y_c + Y_d) \neq \{0\} \tag{J.19}$$

We thus introduce five different variables δ, δ', δ'' etc., representing δ_i, δ_{i+1}, δ_{i+2}, δ_{i-1} and δ_{i-2}, and we formulate the relation as the vanishing of several determinants, as explained in section J.1.1. We then reduce the resulting ideal by factoring out any occurrences of $(\delta - \delta')$, $(\delta - \delta'')$, etc. to assure that δ, δ' etc. are really different solutions. That is, we consider the ideal formed by $F(\delta)$, $F(\delta')$ etc. $(F(\delta) - F(\delta'))/(\delta - \delta')$, etc. and the various determinants derived from J.19, and we apply the Gröbner basis reduction or the Euclidean algorithm to this ideal to find a basis.

Only multiples of some u_i satisfy such a relation (J.18) over $\mathbf{Z}_p$. We fix a multiple of each u_i by normalizing u_i to have first coordinate 1. The relations finally serve to define u_i in terms of δ_i.

By a similar argument, there is a quadratic equation $G(\delta_i, \delta_{i+1})$ expressing δ_{i+1} in terms of δ_i, whose two solutions are δ_{i+1} and δ_{i-1}. For $k > 5$, the algebraic condition is that the corresponding spaces Y_i, Y_{i+1} are in two different triples of subspaces enjoying linear relations:

$$\operatorname{rank}(Y_i + Y_{i+1} + Y_{i+2}) = \operatorname{rank}(Y_i + Y_{i+1} + Y_{i-1}) = 5 \tag{J.20}$$

Special considerations for small k. For $k = 5$, the above arguments do not apply since there are more instances of the relation

$$Y_i \cap (Y_a + Y_b) \cap (Y_c + Y_d) \neq \{0\}$$

with distinct (a, b, c, d, i). For example

$$Y_1 \cap (Y_2 + Y_5) \cap (Y_3 + Y_4)$$

is a one-dimensional space spanned by a vector of the form $u_5 + \mu u_2$ for some constant μ. It turns out that a pair of adjacent spaces such as $(Y_3 + Y_4)$ appear in three such relations whereas a pair of non-adjacent spaces such as $(Y_2 + Y_5)$ appears only in one.This gives an algebraic condition to identify pairs of adjacent δ_is. Once this is done, we can use a two pairs of adjacent indices (a, b), (c, d), with a, b, c, d distinct and different from i and the relation

$$u_i \in Y_i \cap (Y_a + Y_b) \cap (Y_c + Y_d)$$

in order to express u_i as a function of δ_i of degree 4.

The particular case $k = 3$ deserves independent discussions and is postponed until section J.4.

Returning to the general case $k \geq 5$, we represent the solution of the quadratic equation by τ, and say that (δ, τ) generates a pair of 'adjacent' elements (u_i, u_{i+1}) (elements which are multiplied together in the original signature). We think of δ as generating an extension of degree k over $\mathbf{Z}_N$, and τ as generating an extension of degree 2 over $\mathbf{Z}_N[\delta]/F(\delta)$. The ability to distinguish the unordered pairs of 'adjacent' roots $\{\delta_i, \delta_{i+1}\}$ makes the system similar, in spirit, to a Galois extension of $\mathbf{Q}$ whose Galois group is the dihedral group on k elements. We will call on this analogy later. (Remark: it is only an analogy, because δ and τ really are elements of the ground fields.)

We can get the missing kth equation

$$v'_k = \sum_i u_i u_{i+1} \tag{J.21}$$

The coefficients of v'_k in terms of $y_j y_\ell$ ostensibly depend on δ_i and on the pairings (δ_i, δ_{i+1}), or equivalently on (δ, τ). But the coefficients would come out the same no matter which solution (δ, τ) were chosen, that is, no matter whether we assigned the ordering $(1, 2, 3, \ldots, k)$ or $(3, 2, 1, k, k-1, \ldots, 4)$ to the solutions u_i. This means that the coefficients will be in fact independent of (δ, τ). They will be expressible in terms of only the coefficients of the original $v_i, 1 \leq i \leq k$. This is because they are symmetric (up to dihedral symmetry) in the solutions δ_i.

The arguments here are analogous to those of Galois theory. Each coefficient c of v'_k is expressed as

$$c = \sum_{0 \leq i < k, 0 \leq j \leq 1} w_{ij} \delta^i \tau^j \tag{J.22}$$

For each of $2k$ different choices of (δ, τ) the value of c comes out the same. Treating (J.22) as $2k$ linear equations in the $2k$ unknowns w_{ij}, with coefficients given by $\delta^i \tau^j$ for various choices of (δ, τ), we must find (if the matrix has full rank) that $w_{00} = c$, and $w_{ij} = 0$ for $(i, j) \neq (0, 0)$.

Now we wish to solve a particular signature. We are given the integer values $v_1, \ldots, v_{k-1}$, and we assign an arbitrary value to v'_k. We have the equations relating v_i to $u_j u_{j+1}$:

$$v_i = \sum_j b'_{ij} u_j u_{j+1} \tag{J.23}$$

where b'_{ij} depends on δ_j. Select (symbolically) one pair (δ, τ) to fix the first two solutions (u_1, u_2), and compute the others in terms of (δ, τ). Then we have $b'_{ij} u_j u_{j+1}$ depending only on (δ, τ).

At this point we have v'_k (which is a v_k-like quadratic form), and A-like and B-like matrices respectively denoted by A' and B' (expressing linear transformations); the entries of all of these live in the pseudo Galois extension $\mathbf{Z}_N[\delta, \tau]/F(\delta)/G(\delta, \tau)$ expressing linear transformations. All rational operations can be done in this domain so this enables us to sign in it. Since the resulting signature does not depend on the ordering of the δ_i, its coefficients will always be in $\mathbf{Z}_N$. Thus it is possible to forge any signature working in a more complicated domain and getting results which always end up in $\mathbf{Z}_N$.

The attack has been implemented on a Sparc Workstation using the computer algebra system MAPLE. It computes a secret key-like (v'_k, A', B') within few hours and then forges any signature in a negligible time.

Example. As for the first scheme, we give a toy example with $N = 97$, $k = 5$ and $s = 1$. The public key is as follows:

$$\begin{aligned}
v_1 &= 11y_1^2 + 31y_2^2 + 15y_3^2 + 8y_4^2 + 5y_5^2 + 23y_1y_2 + 89y_1y_3 + 60y_1y_4 + \\
&\quad 47y_1y_5 + 43y_2y_3 + 24y_2y_4 + 93y_2y_5 + 9y_3y_4 + 78y_3y_5 + 32y_4y_5 \\
v_2 &= 83y_1^2 + 32y_2^2 + 16y_3^2 + 13y_4^2 + 92y_5^2 + 28y_1y_2 + 83y_1y_3 + 58y_1y_4 + \\
&\quad 84y_1y_5 + 58y_2y_3 + 64y_2y_4 + 84y_2y_5 + 38y_3y_4 + 69y_3y_5 + 36y_4y_5 \\
v_3 &= 45y_1^2 + 33y_2^2 + 96y_3^2 + 75y_4^2 + 90y_5^2 + 34y_1y_2 + 51y_1y_3 + 89y_1y_4 + \\
&\quad 26y_1y_5 + 16y_2y_3 + 90y_2y_4 + 42y_2y_5 + 9y_3y_4 + 8y_3y_5 + 47y_4y_5 \\
v_4 &= 65y_1^2 + 54y_2^2 + 96y_3^2 + 33y_4^2 + 26y_5^2 + 46y_1y_2 + 25y_1y_3 + 75y_1y_4 + \\
&\quad 76y_1y_5 + 59y_2y_3 + 66y_2y_4 + 95y_2y_5 + 69y_3y_4 + 48y_3y_5 + 56y_4y_5
\end{aligned}$$

For the sake of brevity, we did not include the secret key, since we will show how to sign, given the public key only. As far as signature verification

is concerned, we propose, as an example, a valid signature of the message which hashes onto $(1, 2, 3, 4)$, namely.

$$y_1y_2 = 7\ ,\ y_2y_3 = 92\ ,\ y_3y_4 = 69\ ,\ y_4y_5 = 54\ ,\ y_5y_1 = 70$$

From these values, one can compute all corresponding values of y_iy_j and check that $(v_1, v_2, v_3, v_4) = (1, 2, 3, 4)$.

As explained above, we consider the quadratic form

$$v_1 + \delta v_2 + \epsilon_3 v_3 + \epsilon_4 v_4$$

and we express the vanishing of all its (3×3) minors. As an example, one of the determinants provides the following equation:

$$7\delta + 15\epsilon_4 + 83 + 18\delta^2 + 31\delta\epsilon_3 + 69\delta\epsilon_4 + 17\epsilon_3^2 + 71\epsilon_4^2 + 32\epsilon_3\epsilon_4 + 49\delta^2\epsilon_3 + 40\delta^2\epsilon_4 + 47\delta\epsilon_3^2 + 54\delta\epsilon_4^2 + 78\epsilon_3^2\epsilon_4 + \epsilon_3\epsilon_4^2 + 24\epsilon_4^3 + 50\epsilon_3^3 + 38\delta^3 + 67\delta\epsilon_3\epsilon_4 + 5\epsilon_3 = 0.$$

Using reduction, we get:

$$\begin{aligned} F(\delta) &= 92 + 58\delta + 51\delta^2 + 43\delta^3 + 72\delta^4 + \delta^5 \\ \epsilon_3 &= 44 + 29\delta + 83\delta^2 + 95\delta^3 + 56\delta^4 \\ \epsilon_4 &= 87 + 14\delta + 94\delta^2 + 33\delta^3 + 38\delta^4 \end{aligned}$$

Next, we make use of the ideal generated by all $F(\delta)$, $F(\delta')$, $(F(\delta) - F(\delta'))/(\delta - \delta')$ etc. and by the polynomials expressing that two roots are adjacent. Using τ to denote a root adjacent to δ and reducing the ideal, we get, as expected, an equation of degree two w.r.t. τ, say $G(\delta, \tau)$ and we compute the other roots in terms of δ and τ. This yields:

$$\begin{aligned} G(\delta,\tau) &= 67\delta^4\tau + 20\delta^4 + 89\delta^3\tau + 26\delta^3 + 68\delta^2\tau + 85\delta^2 + 6\delta\tau + \\ &\quad 3\delta + \tau^2 + 43\tau + 57 \\ \delta_3 &= 68\delta^4\tau + 23\delta^4 + 90\delta^3\tau + 86\delta^3 + 85\delta^2\tau + 18\delta^2 + 93\delta\tau + \\ &\quad 35\delta + 42\tau + 93 \\ \delta_4 &= 29\delta^4\tau + 44\delta^4 + 7\delta^3\tau + 3\delta^3 + 12\delta^2\tau + 50\delta^2 + 4\delta\tau + 67\delta + \\ &\quad 55\tau + 72 \\ \delta_5 &= 30\delta^4 + 8\delta^3 + 29\delta^2 + 91\delta + 96\tau + 54 \end{aligned}$$

We also compute the values of all normalized u_is in terms of δ and τ from equations (J.18). As an example, here is the output for u_2:

$$\begin{aligned} u_2 = {} & y_1 + \\ & y_2(85\delta^4\tau + 89\delta^4 + 15\delta^3\tau + 87\delta^3 + 38\delta^2\tau + 88\delta^2 + 69\tau\delta + 6\delta + 35\tau + 12) + \\ & y_3(86\delta^4\tau + 31\delta^4 + 41\delta^3\tau + 13\delta^3 + 24\delta^2\tau + 18\delta^2 + 52\delta\tau + 62\delta + 15\tau + 17) + \\ & y_4(43\delta^4\tau + 45\delta^4 + 52\delta^3\tau + 38\delta^3 + 68\delta^2\tau + 58\delta^2 + 2\delta\tau + 88\delta + 27\tau + 87) + \\ & y_5(28\delta^4\tau + 4\delta^4 + 8\delta^3\tau + 75\delta^3 + 74\delta^2\tau + 73\delta^2 + 45\delta + 29\delta\tau + 58\tau + 75). \end{aligned}$$

All computations now take place in the pseudo Galois extension

$$\mathbf{Z}_N[\delta,\tau]/F(\delta)/G(\delta,\tau).$$

We choose v_5 as the sum of all u_iu_{i+1}. As expected, this value turns out to be "independent" of δ and τ:

$$\begin{aligned} v_5 \;=\; & 5y_1^2+30y_2^2+89y_3^2+67y_4^2+5y_5^2+35y_1y_2+4y_1y_3+62y_1y_4+61y_1y_5+ \\ & 32y_2y_3+6y_2y_4+14y_2y_5+13y_3y_4+63y_3y_5+87y_4y_5 \end{aligned}$$

Using elementary linear algebra, we finally compute a B^{-1}-like matrix, which takes the v_i to u_iu_{i+1}, and an A^{-1}-like matrix which computes the y_is from the u_is . Both matrices appear in terms of δ and τ.

Once this precomputation has been done, we can forge any signature. For instance, in order to sign the hashed value $(1,2,3,4)$, we randomly choose $v_5=44$, and we get, using our equations the valid signature:

$$y_1y_2=59\ ,\ y_2y_3=60\ ,\ y_3y_4=38\ ,\ y_4y_5=26\ ,\ y_5y_1=56$$

We note that it is perfectly possible to implement the last step "formally" and to sign a formal message (V_1,V_2,V_3,V_4,V_5), thus recovering a substitute to the original signing function.

J.4 Extensions

J.4.1 Extension to the Case $s>1$

The case $s>1$ is more complicated and we only sketch a possible attack. This part has not been implemented. Suppose again that we have k variables $y_1,y_2,\ldots,y_k$, with k odd, whose pairwise products generate the signature, and that the hashed message has $k-s$ quantities $v_1,v_2,\ldots,v_{k-s}$, together with coefficients $c_{ij\ell}$ expressing v_i in terms of y_jy_ℓ. Suppose for simplicity that $s>1$ is odd, so that $k-s$ is even.

Some linear combinations of the $k-s$ quadratic forms v_i will have rank $s+1$. Namely, for each index set $I\subseteq\{1,2,\ldots,k\}$ of size $(s+1)/2$ such that $\forall i,j\in I\colon \mid i-j\mid\geq 2$ and $\{1,k\}\not\subseteq I$, there is such a linear combination of the form

$$\sum_{i\in I}u_i(\alpha_{iI}u_{i-1}+\beta_{iI}u_{i+1}) \tag{J.24}$$

The number of such index sets I is

$$\frac{k}{\frac{s+1}{2}}\left(\begin{array}{c} k-\frac{s+3}{2} \\ \frac{s-1}{2} \end{array}\right) \tag{J.25}$$

There are more than k linear combinations, leading to increased complication. The space Y_I, spanned by rows of the corresponding quadratic form, contains u_i for each index $i \in I$. So each u_i is in the intersection of a large number of subspaces Y_I, and hopefully only multiples of u_i will be in such an intersection. This algebraic condition should distinguish the u_i, hopefully indexing them by the roots δ of some polynomial $F(\delta)$ of degree k. Pairs $\{u_i, u_{i+2}\}$ of solutions with index differing by 2 should be distinguished by appearing together in many different subspaces Y_I. From this we would be able to distinguish pairs $\{u_i, u_{i+1}\}$. We would fabricate the missing equations as follows: for $j = k - s + 1, \ldots, k$, let $u'_{i(j)}$ be a multiple of u_i, normalized to have a 1 in position j, and set $v'_j = \sum_i u'_{i(j)} u'_{i+1(j)}$.

J.4.2 The Case k=3, s=1

In the special case $k = 3$, $s = 1$, where we must satisfy two quadratic equations in three variables, we can employ an *ad hoc* method, since the methods outlined in section J.3 don't work. We take a linear transformation of the two quadratic equations so that the right-hand side of one equation vanishes; that is, if the given values are v_1 and v_2, we take v_2 times the first equation minus v_1 times the second. This gives a homogeneous quadratic equation in three variables y_1, y_2, y_3:

$$\sum_{ij} c_{ij} y_i y_j = 0 \tag{J.26}$$

The second equation is inhomogeneous:

$$\sum_{ij} d_{ij} y_i y_j = d_0 \tag{J.27}$$

By setting $z_1 = y_1/y_3$, $z_2 = y_2/y_3$ in (J.26), we obtain an inhomogeneous quadratic equation in two variables z_1, z_2. We can easily find an affine change of basis from z_1, z_2 to z'_1, z'_2 which transforms the equation to the form

$$c'_{11} {z'_1}^2 + c'_{12} z'_1 z'_2 + c'_{22} {z'_2}^2 = c'_0 \bmod N \tag{J.28}$$

and a further linear change of variables to z''_1, z''_2 yielding

$$c''_{11} {z''_1}^2 + c''_{22} {z''_2}^2 = c''_0 \bmod N \tag{J.29}$$

which can be solved by the Pollard [126] attack on the Ong-Schnorr-Shamir [117] scheme. We find from this a set of ratios y_j/y_3, and, by extension, a set of ratios $y_i y_j / y_3^2$, satisfying (J.26). Setting $y_3^2 = \lambda$, the second equation (J.27) becomes a linear equation in λ. Thus we find a consistent set of pairwise products $y_i y_j$ satisfying the desired equations (J.26), (J.27).

J.4.3 Open Questions

The birational permutation signature scheme has many instances, of which we have attacked only the first few examples. For a more complex instance of the scheme, the ideas of the present paper will still apply: the trap door conditions lead to algebraic equations on the coefficients of the transformations, and we hope to gather enough such equations to make it possible to solve them by g.c.d. or Gröbner basis methods. But, for any specific instance, it remains to see whether the ideas of the present paper would be sufficient to mount an attack.

One general theme is that when solutions of the algebraic equations enjoy a symmetry, it makes the equations harder to solve, but we don't need to solve them, since the final solution will enjoy the same symmetry, and quantities symmetric in the roots of the equation can be expressed in terms of the coefficients of the equation alone, not in terms of the roots. When the roots fail to enjoy a symmetry, they can be distinguished by algebraic conditions, which yield further algebraic equations, and the Gröbner basis methods have more to work with. This gives us hope that the methods outlined in this paper will apply with some generality to many instances of the birational permutation signature scheme.

Conclusion

We have shown how to use algorithmic tools to break many of the cryptographic schemes based on birational permutations with hidden trap-doors. Though not all the cases proposed by Shamir have been studied, we demonstrated that use of Galois-like theory may break them. This enlightens cryptanalysis with a new approach. We would like to comment briefly on the mathematics of our attacks. We used pseudo-extensions of pseudo-fields to break the most significant proposals. In a way, this is very similar to the security analysis of RSA-like cryptosystems: stated in a provocative way, this security corresponds to the freedom of treating (at least algorithmically) $\mathbf{Z}_N$ as a field since we *do not know* a non-trivial factor of N. In a similar vein, we took the freedom to consider $\mathbf{Z}_N[\delta]/F(\delta)$ as a Galois extension since we *did not know* how to get a root of F. This is a way to use formally incorrect statements of mathematics in order to achieve actual results.

Annexe K

Hidden Collisions on DSS

[*Cet article de* Serge Vaudenay *a été publié dans les actes du colloque Crypto 96 [166].*]

[**Erratum.** Equation (K.1) occurs with probability $\frac{1}{2} \times \frac{1}{4} \times \left(\frac{3}{4}\right)^{158} \approx 2^{-68.58}$, so we need $2^{74.36}$ trials rather than $2^{73.95}$.]

Abstract. We explain how to forge public parameters for the Digital Signature Standard with two known messages which always produce the same set of valid signatures (what we call a collision). This attack is thwarted by using the generation algorithm suggested in the specifications of the Standard, so it proves one always need to check proper generation. We also present a similar attack when using this generation algorithm within a complexity 2^{74}, which is better than the birthday attack which seeks for collisions on the underlying hash function.

Imagine you want to join to a brand new association which offers to provide useful services on the net. To allow electronic payment, this association provides a DSS implementation with public parameters

$$\begin{aligned} p = {} & 10073861752742838167330548434435874326642998021609289724334546391745980774853739798193524368725087200308751842119703988500905836011228131038288614407907 61 \end{aligned}$$

and

$$q = 759902064211816970120975637406935605590678547999.$$

To check the connection, the server requests you to sign the message "`This is just a test`". Then time goes, and your bank warns you that you need to feed your account after your last check of $\$9,302$. Of course, the manager of the association has just disappeared with the money obtained from all the 215 members! (This is a $\$2,000,000$ swindle.)

This attack comes from a special forgery of the public parameter. It is *not* applicable to the accurate DSS suggested in [6] since it requires the production of a certificate of *good* forgery. This attacks however proves that the signer must check the certificate himself (which is not explicitly mentioned is the Standard), which may be a very cumbersome task for low cost devices.

In this paper, we explain how to perform this attack. We also show a similar attack against DSS with the suggested certificate of good forgery with a complexity equivalent to 2^{74} SHS computations. The complexities of the attacks do not depend on the length of the prime p used in the signature scheme.

K.1 Signatures Based on Discrete Logarithm

The first digital signature algorithm based on the discrete logarithm problem (namely, the discrete log problem in the group $\mathbf{Z}_p{}^*$, given a prime number p) was the ElGamal signature [50]. This scheme produces quite long signatures and is also subject to Bleichenbacher's attack which uses small factors of $p-1$ [31].

The Schnorr signature [140, 142] repars those two shortcomings by using an element g whose order is a 160-bit prime factor q of $p-1$. The underlying group for the discrete log problem is thus a subgroup of $\mathbf{Z}_p{}^*$ with order q. The Digital Signature Standard (DSS) [6] also uses such a g.

In the following, we only consider the case of the DSS scheme. In the signature scheme, the message is processed through the Secure Hash Standard (SHS) [5] which produces a 160-bit digest. This value then appears as a power of g. Hence, the *real* hash value is not the output of SHS, but rather the message digest reduced modulo q. Since q is less than the largest output of SHS, this may produce collisions.

For completeness, we now recall the outlines of DSS.

p, q and g are the public parameters chosen by the authority. p is a 512-bit prime (or a 1024-bit prime in stronger versions), q is a 160-bit prime factor of $p-1$, and g is a primitive qth root of 1 modulo p. Each user has a secret key x (which is a 160-bit integer) and publishes a corresponding 512-bit public key $y = g^x \bmod p$. The signature of an arbitrary message m using a (secret)

fresh random 160-bit integer k is a (r, s) pair of 160-bit integers defined by

$$\begin{aligned} r &= (g^k \bmod p) \bmod q \\ s &= \frac{\mathrm{SHS}(m) + xr}{k} \bmod q. \end{aligned}$$

The verification of the signature is performed by checking

$$r = \left(g^{\frac{\mathrm{SHS}(m)}{s} \bmod q} y^{\frac{r}{s} \bmod q} \bmod p \right) \bmod q.$$

K.2 Collision for DSS

We have noticed that the *real* hash function which is used in DSS is SHS mod q. Hence, if we know a pair of messages (m, m') such that

$$\mathrm{SHS}(m) \equiv \mathrm{SHS}(m') \pmod{q}$$

any signature from any user of message m is also a valid signature of message m'. We call such a pair a *collision* for DSS.

Of course, since SHS is considered as a *secure* hash function, it is still infeasible to get a collision. More precisely, the complexity of the best attack (based on the birthday paradox) has a complexity of 2^{80}. Similarly, it may be infeasible, given a 160-bit prime q, to find a collision on SHS mod q. It is however possible to construct q from an (un)willing collision (m, m').

More concretely, from a random pair (m, m'), we can check whether or not $q = |\mathrm{SHS}(m) - \mathrm{SHS}(m')|$ is a 160-bit prime. The integer $|\mathrm{SHS}(m) - \mathrm{SHS}(m')|$ is obviously a 160-bit integer with probability 1/2. Then, a random 160-bit integer is a prime with probability approximately $1/160 \log 2 \approx 1/111$, thanks to the Prime Number Theorem. Hence, with an average of 222 trials, we obtain a collision which defines a valid prime q. It is not difficult, given a prime q, to issue valid p and g, for instance by following the generation algorithm provided in [6].

The example given in the introduction uses the message

$$m = \texttt{This is just a test}$$

which is encoded as

54686973	20697320	6a757374	20612074
65737480	00000000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000098

in hexadecimal and its hash value is

$$\mathrm{SHS}(m) = 653095248274681173784642234520335115855431883588.$$

The second message is

$$m' = \texttt{Transfer \$9,302 on account XYZ}$$

and its hash value is

$$\mathrm{SHS}(m') = 141299731248649814390561787192727072144611043158 7.$$

So let

$$q = \mathrm{SHS}(m) - \mathrm{SHS}(m')$$

which is a 160-bit prime. Then one can issue a 512-bit prime p such that q divides $p - 1$.

K.3 The Public Parameters Generation

The description of DSS suggests that q is first generated from a random seed by a specific algorithm, then p and g are issued from q by the same seed. The seed should be used as a certificate of a honest forgery for p and q. The previous attack suggests that this feature must be used.

Is the attack really thwarted by this generation algorithm? We recall that q is set to

$$(\mathrm{SHS}(seed) \oplus \mathrm{SHS}(seed+1)) \vee 2^{159} \vee 1$$

until it is a prime where $\oplus$ and $\vee$ denote the bitwise exclusive and inclusive or respectively. (We notice that we need in average about 55 trials to get a prime q since this is a random 160-bit odd integer.) Let $seed = m$ and $seed+1 = m'$. The same attack holds whenever

$$|\mathrm{SHS}(seed) - \mathrm{SHS}(seed+1)| = (\mathrm{SHS}(seed) \oplus \mathrm{SHS}(seed+1)) \vee 2^{159} \vee 1 \quad \text{(K.1)}$$

Since this occurs with probability $\frac{1}{2} \times \frac{1}{4} \times \left(\frac{3}{4}\right)^{157} \approx 2^{-68.16}$, we obtain that $2^{73.95}$ trials are required on average to mount this attack.

The Standard says that parameters p and q shall be generated as suggested above, or using other FIPS approved security methods. This study may thus be helpful for proposing other generation algorithms. For instance, we can suggest to set

$$q = \mathrm{SHS}(seed) \vee 2^{159} \vee 1$$

until q is valid.

K.4 On the g Parameter

Surprisingly, no particular care is required for parameter g. Thus, fake gs like $g = 0$ or 1 are implicitly accepted! Actually, a dishonest authority can provide $g = 1$ to a user and make him accept any signature forged by

$$\begin{aligned} r &= (y^k \bmod p) \bmod q \\ s &= \frac{r}{k} \bmod q. \end{aligned}$$

In a more dedicated attack, the authority provides $g = y^\alpha \bmod p$ to Alice, where y is the public key of Bob, and forges Bob's signatures by

$$\begin{aligned} r &= (y^k \bmod p) \bmod q \\ s &= \frac{\alpha \mathrm{SHS}(m) + r}{k} \bmod q. \end{aligned}$$

We thus suggest to add a certificate of valid forgery of g when issuing the public parameters.

K.5 Conclusion

We performed an attack against DSS which allows the issuing autority to forge valid public parameters with hidden collisions. It confirms internal problems in the signature scheme which were already predicted by Pointcheval and Stern in [124] because of not using a random pattern in the hash function like in the Schnorr signature [140]. We have proved that when the issuing authority is not trusted, all users must then check proper generation of the public parameters. We also showed how to adapt this attack to the generation algorithm suggested in the Standard within a complexity 2^{74}. Even if the complexity were smaller, the attack would still not be so dramatic because it is easy to detect. This should be registered as an existing (bad) property of DSS though.

This attack can easily be avoided by using a 161-bit prime q, or by dropping the most significant bit of SHS (or by setting the least significant bit to a constant before using it in the signature scheme), or by padding a random pattern before hashing. Its complexity can also be increased up to 2^{80} by using a stronger certificate-based generation algorithm for p and q.

We also presented attacks based on malicious forgery of the g parameter. We thus recommand to use a suitable certificate of honest forgery for g as well as for p and q.

Bibliographie

[1] *ANSI X3.106–1983*, American National Standard for Information Systems — Data Encryption Algorithm — Modes of Operations. American National Standards Institute, New York, 1983.

[2] *FIPS 46*, Data Encryption Standard. U.S. Department of Commerce — National Bureau of Standards, National Technical Information Service, Springfield, Virginia. *Federal Information Processing Standard Publication 46*, 1977.

[3] *FIPS 81*, DES Modes of Operation. U.S. Department of Commerce — National Bureau of Standards, National Technical Information Service, Springfield, Virginia. Federal Information Processing Standards 81, 1980.

[4] *FIPS 180*, Secure Hash Standard. U.S. Department of Commerce — National Institute of Standards and Technology, National Technical Information Service, Springfield, Virginia. Federal Information Processing Standards 180, 1993.

[5] *FIPS 180-1*, Secure Hash Standard. U.S. Department of Commerce — National Institute of Standards and Technology, National Technical Information Service, Springfield, Virginia. Federal Information Processing Standards 180-1, 1995.

[6] *FIPS 186*, Digital Signature Standard. U.S. Department of Commerce — National Institute of Standards and Technology, National Technical Information Service, Springfield, Virginia. Federal Information Processing Standards 186, 1994.

[7] *ISO 8732*, Banking — Key Management (Wholesale). International Organization for Standardization, Geneva, Switzerland, 1988.

[8] *ISO/IEC 10116*, Information Processing — Modes of Operation for an n-bit Block Cipher Algorithm. International Organization for Standardization, Geneva, Switzerland, 1991.

[9] Organisation for Economic Co-operation and Development *Cryptography Policy Guidelines*, 27 March, 1997.

[10] Méthode de Chiffrement Fondée sur la Décorrélation. In *100 Faits Marquants du Département des Sciences Pour l'Ingénieur*, p. 15, CNRS, 1997.

[11] C. M. Adams. *A Formal and Practical Design Procedure for Substitution-Permutation Network Cryptosystems.*, Ph.D. Thesis of Queen's University, Kingston, Ontario, Canada, 1990.

[12] C. M. Adams, S. E. Tavares. Designing s-boxes Resistant to Differential Cryptanalysis. In *Proceedings of 3rd Symposium on the State and Progress of Research in Cryptography*, pp. 386–397, Rome, Italy, 1994.

[13] N. Alon. Eigenvalues and Expanders. in *Combinatorica*, vol. 6, pp. 83–96, 1986.

[14] N. Alon, V. D. Milman. λ_1, Isoperimetric Inequalities for Graphs, and Superconcentrators. In *Journal of Combinatorical Theory*, vol. B 38, pp. 73–88, 1985.

[15] R. J. Anderson. The Classification of Hash Functions. In *Proceedings of the 4th IMA Conference on Cryptography and Coding*, Cirencester, United Kingdom, pp. 83–95, Oxford University Press, 1995.

[16] R. Anderson, S. Vaudenay. Minding your p's and q's. In *Advances in Cryptology ASIACRYPT'96*, Kyongju, Corée, Lecture Notes in Computer Science 1163, pp. 26–35, Springer-Verlag, 1996.

[17] R. Anderson, S. Vaudenay, B. Preneel, K. Nyberg. The Newton channel. In *Proceedings of the First International Workshop on Information Hiding*, Cambridge, Royaume Uni, Lecture Notes in Computer Science 1174, pp. 151–156, Springer-Verlag, 1996.

[18] K. Aoki, K. Ohta. Strict Evaluation of the Maximum Average of Differential Probability and the Maximum Average of Linear Probability. *IEICE Transactions on Fundamentals*, vol. E80-A, pp. 1–8, 1997.

[19] L. Babai, M. Szegedy. Local Expansion of Symmetrical Graphs. In *Combinatorics, Probability and Computing*, vol. 1, pp. 1–11, 1992.

[20] T. Baritaud, H. Gilbert, M. Girault. FFT Hashing is not Collision-Free. In *Advances in Cryptology EUROCRYPT'92*, Balatonfüred, Hongrie, Lecture Notes in Computer Science 658, pp. 35–44, Springer-Verlag, 1993.

[21] M. Bellare and P. Rogaway. Random Oracles are Practical: a Paradigma for Designing Efficient Protocols. In *1st ACM Conference on Computer and Communications Security*, Fairfax, Virginie, U.S.A., pp. 62–73, ACM Press, 1993.

[22] I. Ben-Aroya, E. Biham. Differential Cryptanalysis of Lucifer. *Journal of Cryptology*, vol. 9, pp. 21–34, 1996.

[23] T. A. Berson, L. R. Knudsen. Truncated Differentials on Safer. In *Fast Software Encryption'96*, Cambridge, Royaume Uni, Lecture Notes in Computer Science 1039, pp. 15–26, Springer-Verlag, 1996.

[24] E. Biham. On Matsui's Linear Cryptanalysis. In *Advances in Cryptology EUROCRYPT'94*, Pérouse, Italie, Lecture Notes in Computer Science 950, pp. 341–355, Springer-Verlag, 1995.

[25] E. Biham. New Types of Cryptanalytic Attacks using Related Keys. *Journal of Cryptology*, vol. 7, pp. 229–246, 1994.

[26] E. Biham. A Fast new DES Implementation in Software. In *Fast Software Encryption'97*, Haifa, Israel, Lecture Notes in Computer Science 1267, pp. 260–272, Springer-Verlag, 1997.

[27] E. Biham, A. Shamir. Differential Cryptanalysis of DES-like Cryptosystems. In *Advances in Cryptology CRYPTO'90*, Santa Barbara, Californie, U.S.A., Lecture Notes in Computer Science 537, pp. 2–21, Springer-Verlag, 1991.

[28] E. Biham, A. Shamir. Differential Cryptanalysis of DES-like Cryptosystems. *Journal of Cryptology*, vol. 4, pp. 3–72, 1991.

[29] E. Biham, A. Shamir. Differential Cryptanalysis of the Full 16-Round DES. In *Advances in Cryptology CRYPTO'92*, Santa Barbara, Californie, U.S.A., Lecture Notes in Computer Science 740, pp. 487–496, Springer-Verlag, 1993.

[30] E. Biham, A. Shamir. *Differential Cryptanalysis of the Data Encryption Standard*, Springer-Verlag, 1993.

[31] D. Bleichenbacher. Generating ElGamal Signatures without Knowing the Secret Key. In *Advances in Cryptology EUROCRYPT'96*, Saragosse, Espagne, Lecture Notes in Computer Science 1070, pp. 10–18, Springer-Verlag, 1996.

[32] B. den Boer, A. Bosselaers. An Attack on the Last two Rounds of MD4. In *Advances in Cryptology CRYPTO'91*, Santa Barbara, Californie, U.S.A., Lecture Notes in Computer Science 576, pp. 194–203, Springer-Verlag, 1992.

[33] P. Camion, H. Chabanne. On the Powerline System. In *Advances in Cryptology, ICICS'97*, Beijing, China, Lecture Notes in Computer Science 1334, pp. 381–385, Springer-Verlag, 1997.

[34] L. Carter, M. Wegman. Universal Classes of Hash Functions. *Journal of Computer and System Sciences*, vol. 18, pp. 143–154, 1979.

[35] F. Chabaud, S. Vaudenay. Links Between Differential and Linear Cryptanalysis. In *Advances in Cryptology EUROCRYPT'94*, Pérouse, Italie, Lecture Notes in Computer Science 950, pp. 356–365, Springer-Verlag, 1995.

[36] B. Chor, R. L. Rivest. A Knapsack-Type Public Key Cryptosystem based on Arithmetic in Finite Fields. In *Advances in Cryptology CRYPTO'84*, Santa Barbara, Californie, U.S.A., Lecture Notes in Computer Science 196, pp. 54–65, Springer-Verlag, 1985.

[37] B. Chor, R. L. Rivest. A Knapsack-Type Public Key Cryptosystem based on Arithmetic in Finite Fields. *IEEE Transactions on Information Theory*, vol. IT-34, pp. 901–909, 1988.

[38] D. Coppersmith. The Data Encryption Standard (DES) and its Strength against Attacks. *IBM Journal of Research and Development*, vol. 38, pp. 243–250, 1994.

[39] D. Coppersmith, J. Stern and S. Vaudenay. Attacks on the Birational Permutation Signature Schemes. In *Advances in Cryptology CRYPTO'93*, Santa Barbara, Californie, U.S.A., Lecture Notes in Computer Science 773, pp. 587–593, Springer-Verlag, 1994.

[40] D. Coppersmith, J. Stern, S. Vaudenay. The security of the birational permutation signature schemes. *Journal of Cryptology*, vol. 10, pp. 207–221, 1997. (Voir annexe J.)

[41] H. Cramer. *Mathematical Methods of Statistics*, Princeton University Press, 1946.

[42] S. R. Czapor, K. O. Geddes and G. Labahn. *Algorithms for Computer Algebra*, Kluwer Academic Press, 1992.

[43] I. B. Damgård. A Design Principle for Hash Functions. In *Advances in Cryptology CRYPTO'89*, Santa Barbara, Californie, U.S.A., Lecture Notes in Computer Science 435, pp. 416–427, Springer-Verlag, 1990.

[44] R. W. Davies, W. L. Price. Digital Signature – an Update. In *Proceedings of the International Conference on Computer Communications*, Sydney, pp. 843-847, North-Holland, 1985.

[45] J. Dénes, A. D. Keedwell. *Latin Squares and their Applications*, Akadémiai Kiadó, Budapest, 1974.

[46] W. Diffie. The First Ten Years of Public Key Cryptology. In *Contemporary Cryptology — The Science of Information Integrity*, G. J. Simmons Editor, pp. 135–175, IEEE Press, 92.

[47] W. Diffie, M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, vol. IT-22, pp. 644–654, 1976.

[48] H. Dobbertin. Cryptanalysis of MD4. In *Fast Software Encryption'96*, Cambridge, Royaume Uni, Lecture Notes in Computer Science 1039, pp. 53–69, Springer-Verlag, 1996.

[49] H. Dobbertin. Almost Perfect nonlinear power functions on GF(2^n). *IEEE Trans. Inf. Theory*, submitted.

[50] T. ElGamal. A Public Key Cryptosystem and a Signature Scheme based on Discrete Logarithms. In *IEEE Transactions on Information Theory*, vol. IT-31, pp. 469–472, 1985.

[51] H. Feistel. Cryptography and Computer Privacy. *Scientific American*, vol. 228, pp. 15–23, 1973.

[52] W. Feller. *An Introduction to Probability Theory and its Applications*, vol. 1, Wiley, 1957.

[53] H. Gilbert. *Cryptanalyse Statistique des Algorithmes de Chiffrement et Sécurité des Schémas d'Authentification*, Thèse de Doctorat de l'Université de Paris 11, 1997.

[54] H. Gilbert, G. Chassé. A Statistical Attack of the FEAL-8 Cryptosystem. In *Advances in Cryptology CRYPTO'90*, Santa Barbara, Californie, U.S.A., Lecture Notes in Computer Science 537, pp. 22–33, Springer-Verlag, 1991.

[55] H. Gilbert, M. Girault, P. Hoogvorst, F. Noilhan, T. Pornin, G. Poupard, J. Stern, S. Vaudenay. Decorrelated Fast Cipher: an AES Candidate. (Extended Abstract.) In *Proceedings from the First Advanced Encryption Standard Candidate Conference*, National Institute of Standards and Technology (NIST), August 1998. (Voir annexe H.)

[56] H. Gilbert, M. Girault, P. Hoogvorst, F. Noilhan, T. Pornin, G. Poupard, J. Stern, S. Vaudenay. Decorrelated Fast Cipher: an AES Candidate. Submitted to the Advanced Encryption Standard process. In *CD-ROM "AES CD-1: Documentation"*, National Institute of Standards and Technology (NIST), August 1998.

[57] S. Halevi, H. Krawczyk. MMH: Software Message Authentication in the Gbit/Second Rates. In *Fast Software Encryption'97*, Haifa, Israel, Lecture Notes in Computer Science 1267, pp. 172–189, Springer-Verlag, 1997.

[58] M. Hall, L. J. Paige. Complete Mappings of Finite Groups. In *Pacific Journal of Mathematics*, vol. 5, pp. 541–549, 1955.

[59] C. Harpes. *Partitioning Cryptanalysis*, Post-diploma Thesis, ISI, ETH Zürich, 1994.

[60] C. Harpes, G. G. Kramer, J. L. Massey. A Generalization of Linear Cryptanalysis and the Applicability of Matsui's Piling-up Lemma. In *Advances in Cryptology EUROCRYPT'95*, Saint-Malo, France, Lecture Notes in Computer Science 921, pp. 24–38, Springer-Verlag, 1995.

[61] P. Hawkes. Differential-Linear Weak Key Classes of IDEA. In *Advances in Cryptology EUROCRYPT'98*, Espoo, Finlande, Lecture Notes in Computer Science 1403, pp. 112–126, Springer-Verlag, 1998.

[62] M. E. Hellman, R. Merkle, R. Schroeppel, L. Washington, W. Diffie, S. Pohlig, P. Schweitzer. *Results of an Initial Attempt to Cryptanalyze the NBS Data Encryption Standard*, Stanford University, September 1976.

[63] H. M. Heys. *The Design of Substitution-Permutation Network Ciphers Resistant to Cryptanalysis*, Ph.D. Thesis of Queen's University, Kingston, Ontario, Canada, 1994.

[64] H. M. Heys, S. E. Tavares. Substitution-Permutation Networks Resistant to Differential and Linear Cryptanalysis. *Journal of Cryptology*, vol. 9, pp. 1–19, 1996.

[65] A. Hodges. *Alan Turing: The Enigma of Intelligence*, Unwin Paperbacks, 1985.

[66] K. Huber. Specialised Attack on Chor-Rivest Public Key Cryptosystem. *Electronics Letters*, vol. 27, no. 23, pp. 2130, 1991.

[67] T. Jakobsen, L. R. Knudsen. The Interpolation Attack on Block Ciphers. In *Fast Software Encryption'97*, Haifa, Israel, Lecture Notes in Computer Science 1267, pp. 28–40, Springer-Verlag, 1997.

[68] A. Joux, J. Stern. Lattice Reduction: a Toolbox for the Cryptanalyst. To appear in *Journal of Cryptology*.

[69] M. Just, S. Vaudenay. Authenticated multi-party key agreement. In *Advances in Cryptology ASIACRYPT'96*, Kyongju, Corée, Lecture Notes in Computer Science 1163, pp. 36–49, Springer-Verlag, 1996.

[70] D. Kahn. *The Codebreakers — The Comprehensive History of Secret Communication from Ancient Time to the Internet*, 2nd Edition, Scribner, New York, 1996.

[71] B. R. Kaliski Jr., M. J. B. Robshaw. Linear Cryptanalysis using Multiple Approximations. In *Advances in Cryptology CRYPTO'94*, Santa Barbara, Californie, U.S.A., Lecture Notes in Computer Science 839, pp. 26–39, Springer-Verlag, 1994.

[72] J. B. Kam, G. I. Davida. Structured Design of Substitution-Permutation Encryption Networks. *IEEE Transactions on Computers*, vol. C-28, pp. 747–753, 1979.

[73] A. Kerckhoffs. *La Cryptographie Militaire*, Paris, Librairie Militaire Baudoin & Cie, 1883.

[74] L. R. Knudsen. *Block Ciphers — Analysis, Design and Applications*, Aarhus University, 1994.

[75] L. R. Knudsen. A Key-Schedule Weakness in SAFER K-64. In *Advances in Cryptology CRYPTO'95*, Santa Barbara, Californie, U.S.A., Lecture Notes in Computer Science 963, pp. 274–286, Springer-Verlag, 1995.

[76] N. Koblitz. *A Course in Number Theory and Cryptography*, 2nd Edition, Graduate Texts in Mathematics 114, Springer-Verlag, 1994.

[77] H. Krawczyk LFSR-based Hashing and Authentication. In *Advances in Cryptology CRYPTO'94*, Santa Barbara, Californie, U.S.A., Lecture Notes in Computer Science 839, pp. 129–139, Springer-Verlag, 1994.

[78] K. Kusuda, T. Matsumoto. A Strangth Evaluation of the Data Encryption Standard. Submitted to ISO/TC68, 1996.

[79] X. Lai. *On the Design and Security of Block Ciphers*, ETH Series in Information Processing, vol. 1, Hartung-Gorre Verlag Konstanz, 1992.

[80] X. Lai, J. L. Massey. A Proposal for a New Block Encryption Standard. In *Advances in Cryptology EUROCRYPT'90*, Aarhus, Danemark, Lecture Notes in Computer Science 473, pp. 389–404, Springer-Verlag, 1991.

[81] X. Lai, J. L. Massey, S. Murphy. Markov Ciphers and Differential Cryptanalysis. In *Advances in Cryptology EUROCRYPT'91*, Brighton, Royaume Uni, Lecture Notes in Computer Science 547, pp. 17–38, Springer-Verlag, 1991.

[82] S. Lang. *Algebra*. Second Editon. Addison-Wesley, 1984.

[83] J. Lee, H. M. Heys, S. E. Tavares. On the Resistance of the CAST Encryption Algorithm to Differential Cryptanalysis. Presented at the SAC'95 conference.

[84] H. W. Lenstra, Jr. On the Chor-Rivest Knapsack Cryptosystem. *Journal of Cryptology*, vol. 3, pp. 149–155, 1991.

[85] A. K. Lenstra, H. W. Lenstra Jr., L. Lovász. Factoring Polynomials with Rational Coefficients. *Math. Ann.*, vol. 261, pp. 515–534, 1982.

[86] M. Luby, C. Rackoff. How to Construct Pseudorandom Permutations from Pseudorandom Functions. *SIAM Journal on Computing*, vol. 17, pp. 373–386, 1988.

[87] J. L. Massey. SAFER K-64: a Byte-Oriented Block-Ciphering Algorithm. In *Fast Software Encryption'93*, Cambridge, Royaume Uni, Lecture Notes in Computer Science 809, pp. 1–17, Springer-Verlag, 1994.

[88] J. L. Massey. SAFER K–64: One Year Later. In *Fast Software Encryption'93*, Cambridge, Royaume Uni, Lecture Notes in Computer Science 809, pp. 212–241, Springer-Verlag, 1994.

[89] M. Matsui. Linear Cryptanalysis Methods for DES Cipher. In *Advances in Cryptology EUROCRYPT'93*, Lofthus, Norvège, Lecture Notes in Computer Science 765, pp. 386–397, Springer-Verlag, 1994.

[90] M. Matsui. On Correlation Between the Order of S-boxes and the Strength of DES. In *Advances in Cryptology EUROCRYPT'94*, Pérouse, Italie, Lecture Notes in Computer Science 950, pp. 366–375, Springer-Verlag, 1995.

[91] M. Matsui. The First Experimental Cryptanalysis of the Data Encryption Standard. In *Advances in Cryptology CRYPTO'94*, Santa Barbara, Californie, U.S.A., Lecture Notes in Computer Science 839, pp. 1–11, Springer-Verlag, 1994.

[92] M. Matsui. New Structure of Block Ciphers with Provable Security against Differential and Linear Cryptanalysis. In *Fast Software Encryption'96*, Cambridge, Royaume Uni, Lecture Notes in Computer Science 1039, pp. 205–218, Springer-Verlag, 1996.

[93] M. Matsui. New Block Encryption Algorithm MISTY. In *Fast Software Encryption'97*, Haifa, Israel, Lecture Notes in Computer Science 1267, pp. 54–68, Springer-Verlag, 1997.

[94] F. J. McWilliams, N. J. A. Sloane. *The Theory of Error-correcting Codes*, North-Holland, 1977.

[95] T. Matsumoto and H. Imai. Public Quadratic Polynomial-tuples for Efficient Signature-Verification and Message-Encryption. In *Advances in Cryptology EUROCRYPT'88*, Davos, Suisse, Lecture Notes in Computer Science 330, pp. 419–453, Springer-Verlag, 1988.

[96] S. M. Matyas, C. H. Meyer, J. Oseas. Generating Strong One-way Functions with Cryptographic Algorithm. *IBM Technical Disclosure Bulletin*, vol. 27, pp. 5658–5659, 1985.

[97] U. M. Maurer, J. L. Massey. Local Randomness in Pseudorandom Sequences. *Journal of Cryptology*, vol. 4, pp. 135–149, 1991.

[98] K. McCurley. Cryptographic Number Theory — Ignorance is Bliss. Presentation given at the DIMACS Institute, Rutgers University, 1997.

[99] F. Mébarki. Secret de Deux, Secret de Dieu. *Le journal du CNRS*, num. 94, p. 24, Oct 1997.

[100] W. Meier, O. Staffelbach. Nonlinearity Criteria for Cryptographic Functions. In *Advances in Cryptology EUROCRYPT'89*, Houthalen, Belgique, Lecture Notes in Computer Science 434, pp. 549–562, Springer-Verlag, 1990.

[101] A. J. Menezes, P. C. van Oorschot, S. A. Vanstone. *Handbook of Applied Cryptography*, CRC Press, 1997.

[102] R. C. Merkle. One way Hash Functions and DES. In *Advances in Cryptology CRYPTO'89*, Santa Barbara, Californie, U.S.A., Lecture Notes in Computer Science 435, pp. 416–427, Springer-Verlag, 1990.

[103] R. C. Merkle, M. Hellman. Hiding Information and Signatures in Trap-Door Knapsacks. *IEEE Transactions on Information Theory*, vol. IT-24, pp. 525–530, 1978.

[104] D. M'Raïhi, D. Naccache, J. Stern, S. Vaudenay. Procédé de cryptographie à clé publique basé sur le logarithme discret. Brevet en France num. 95 11622 demandé le 03/10/1995.

[105] D. M'Raïhi, D. Naccache, J. Stern, S. Vaudenay. xmx - a firmware-oriented block cipher based on modular multiplications. In *Fast Software Encryption'97*, Haifa, Israel, Lecture Notes in Computer Science 1267, pp. 166–171, Springer-Verlag, 1997.

[106] D. M'Raïhi, S. Vaudenay. Procédé de cryptographie de messages transmis par un support d'informations à un système de traitement. Brevet en France num. 92 11752 demandé le 02/09/92 (accepté le 25/11/94).

[107] S. Murphy, F. Piper, M. Walker, P. Wild. Likehood Estimation for Block Cipher Keys. Unpublished.

[108] D. Naccache, D. M'Raïhi, D. Raphaeli, S. Vaudenay. Complexity trade-offs with the Digital Signature Standard. In *Advances in Cryptology EUROCRYPT'94*, Pérouse, Italie, Lecture Notes in Computer Science 950, pp. 77–85, Springer-Verlag, 1995.

[109] V. I. Nechaev. Complexity of a Determinate Algorithm for the Discrete Logarithm. In *Mathematical Notes*, vol. 55(2), pp. 165–172, 1994.

[110] K. Nyberg. Perfect Nonlinear S-Boxes. In *Advances in Cryptology EUROCRYPT'91*, Brighton, Royaume Uni, Lecture Notes in Computer Science 547, pp. 378–385, Springer-Verlag, 1991.

[111] K. Nyberg. Differentially Uniform Mapping for Cryptography. In *Advances in Cryptology EUROCRYPT'93*, Lofthus, Norvège, Lecture Notes in Computer Science 765, pp. 55–64, Springer-Verlag, 1994.

[112] K. Nyberg. Linear Approximation of Block Ciphers. In *Advances in Cryptology EUROCRYPT'94*, Pérouse, Italie, Lecture Notes in Computer Science 950, pp. 439–444, Springer-Verlag, 1995.

[113] K. Nyberg. Generalized Feistel Networks. In *Advances in Cryptology ASIACRYPT'96*, Kyongju, Corée, Lecture Notes in Computer Science 1163, pp. 91–104, Springer-Verlag, 1996.

[114] K. Nyberg, L. R. Knudsen. Provable Security against a Differential Cryptanalysis. In *Advances in Cryptology CRYPTO'94*, Santa Barbara, Californie, U.S.A., Lecture Notes in Computer Science 839, pp. 566–574, Springer-Verlag, 1994.

[115] K. Nyberg, L. R. Knudsen. Provable Security against a Differential Cryptanalysis. *Journal of Cryptology*, vol. 8, pp. 27–37, 1995.

[116] L. O'Connor. Properties of Linear Approximation Tables. In *Fast Software Encryption'94*, Leuven, Belgique, Lecture Notes in Computer Science 1008, pp. 131–136, Springer-Verlag, 1995.

[117] H. Ong, C. P. Schnorr and A. Shamir. A fast Signature Scheme based on Quadratic Equations. Proc. 16th ACM Symp. Theory of Computing, pp. 208–216, 1984.

[118] J. Patarin. *Etude des Générateurs de Permutations Basés sur le Schéma du D.E.S.*, Thèse de Doctorat de l'Université de Paris 6, 1991.

[119] J. Patarin. How to Construct Pseudorandom and Super Pseudorandom Permutations from One Single Pseudorandom Function. In *Advances in Cryptology EUROCRYPT'92*, Balatonfüred, Hongrie, Lecture Notes in Computer Science 658, pp. 256–266, Springer-Verlag, 1993.

[120] J. Patarin. Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt '88. In *Advances in Cryptology CRYPTO'95*, Santa Barbara, Californie, U.S.A., Lecture Notes in Computer Science 963, pp. 248–261, Springer-Verlag, 1995.

[121] J. Patarin. About Feistel Schemes with Six (or More) Rounds. In *Fast Software Encryption'98*, Paris, France, Lecture Notes in Computer Science 1372, pp. 103–121, Springer-Verlag, 1998.

[122] J. Pieprzyk. How to Construct Pseudorandom Permutations from a Single Pseudorandom Functions. In *Advances in Cryptology EUROCRYPT'90*, Aarhus, Danemark, Lecture Notes in Computer Science 473, pp. 140–150, Springer-Verlag, 1991.

[123] S. Pohlig, M. Hellman. An Improved Algorithm for Computing Logarithms over GF(q) and its Cryptographic Significance. *IEEE Transactions on Information Theory*, vol. IT-24, pp. 106–110, 1978.

[124] D. Pointcheval, J. Stern. Security Proofs for Signature Schemes. In *Advances in Cryptology EUROCRYPT'96*, Saragosse, Espagne, Lecture Notes in Computer Science 1070, pp. 387–398, Springer-Verlag, 1996.

[125] D. Pointcheval, S. Vaudenay. On Provable Security for Digital Signature Algorithms. Rapport LIENS 96-17, Ecole Normale Supérieure, 1996.

[126] J. M. Pollard and C. P. Schnorr. An Efficient Solution of the Congruence $x^2 + ky^2 = m \pmod{n}$. IEEE Trans. Inform. Theory vol IT-33 no 5, pp. 702–709, 1987.

[127] G. Poupard, S. Vaudenay. Decorrelated Fast Cipher: an AES Candidate well suited for Low Cost Smart Cards Applications. To appear in Cardis 98, LNCS.

[128] R. L. Rivest. The MD4 Message Digest Algorithm. In *Advances in Cryptology CRYPTO'90*, Santa Barbara, Californie, U.S.A., Lecture Notes in Computer Science 537, pp. 303–311, Springer-Verlag, 1991.

[129] R. L. Rivest. The MD4 Message Digest Algorithm. RFC 1186, Oct 1990.

[130] R. L. Rivest. The MD4 Message Digest Algorithm. RFC 1320, Apr 1992.

[131] R. L. Rivest. The MD5 Message Digest Algorithm. RFC 1321, Apr 1992.

[132] R. L. Rivest, A. Shamir and L. M. Adleman. A Method for Obtaining Digital Signatures and Public-key Cryptosystem. In *Communications of the ACM*, vol. 21, pp. 120–126, 1978.

[133] O. S. Rothaus. On Bent Functions. *Journal of Combinatorial Theory*, vol. A20, pp. 300–305, 1976.

[134] R. Rueppel. *Analysis and Design of Stream Ciphers*, Springer-Verlag, 1986.

[135] R. Rueppel. Stream Ciphers. In *Contemporary Cryptology — The Science of Information Integrity*, G. J. Simmons Editor, pp. 65–134, IEEE Press, 92.

[136] B. Schneier. Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish). In *Fast Software Encryption'93*, Cambridge, Royaume Uni, Lecture Notes in Computer Science 809, pp. 191–204, Springer-Verlag, 1994.

[137] B. Schneier. The Blowfish Encryption Algorithm. In *Dr Dobb's Journal*, pp. 38–40, April 1994.

[138] B. Schneier. *Applied Cryptography*, 2nd Edition, John Wiley & Sons, 1996.

[139] B. Schneier, J. Kelsey. Unbalanced Feistel Networks and Block Cipher Design. In *Fast Software Encryption'96*, Cambridge, Royaume Uni, Lecture Notes in Computer Science 1039, pp. 121–144, Springer-Verlag, 1996.

[140] C. P. Schnorr. Efficient Identification and Signature for Smart Cards. In *Advances in Cryptology CRYPTO'89*, Santa Barbara, Californie, U.S.A., Lecture Notes in Computer Science 435, pp. 239–252, Springer-Verlag, 1990.

[141] C. P. Schnorr. FFT-Hashing: an Efficient Cryptographic Hash Function. Présenté à CRYPTO'91. Non publié.

[142] C. P. Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, vol. 4, pp. 161–174, 1991.

[143] C. P. Schnorr. FFT-Hash II, Efficient Cryptographic Hashing. In *Advances in Cryptology EUROCRYPT'92*, Balatonfüred, Hongrie, Lecture Notes in Computer Science 658, pp. 45–54, Springer-Verlag, 1993.

[144] C. P. Schnorr, H. H. Hörner. Attacking the Chor-Rivest Cryptosystem by improved lattice reduction. In *Advances in Cryptology EUROCRYPT'95*, Saint-Malo, France, Lecture Notes in Computer Science 921, pp. 1–12, Springer-Verlag, 1995.

[145] C. P. Schnorr, S. Vaudenay. Parallel FFT-Hashing. In *Fast Software Encryption'93*, Cambridge, Royaume Uni, Lecture Notes in Computer Science 809, pp. 149–156, Springer-Verlag, 1994.

[146] C. P. Schnorr, S. Vaudenay. Black Box Cryptanalysis of Hash Networks based on Multipermutations. In *Advances in Cryptology EUROCRYPT'94*, Pérouse, Italie, Lecture Notes in Computer Science 950, pp. 47–57, Springer-Verlag, 1995.

[147] C. P. Schnorr, S. Vaudenay. Black Box Model for Cryptographic Primitives. *Journal of Cryptology*, vol. 11, pp. 125–140, 1998. (Voir annexe E.)

[148] A. Shamir. A Polynomial Time Algorithm for Breaking the Basic Merkle-Hellman Cryptosystem. In *Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science*, Chicago, Illinois, U.S.A., pp. 145–152, IEEE, 1982.

[149] A. Shamir. Efficient signature schemes based on birational permutations. In *Advances in Cryptology CRYPTO'93*, Santa Barbara, Californie, U.S.A., Lecture Notes in Computer Science 773, pp. 1–12, Springer-Verlag, 1994.

[150] C. E. Shannon. Communication Theory of Secrecy Systems. *Bell system technical journal*, vol. 28, pp. 656–715, 1949.

[151] V. Shoup. Lower Bounds for Discrete Logarithms and Related Problems. In *Advances in Cryptology EUROCRYPT'97*, Constance, Allemagne, Lecture Notes in Computer Science 1233, pp. 256–266, Springer-Verlag, 1997.

[152] M. E. Smid, D. K. Branstad. The Data Encryption Standard: Part and Future. In *Contemporary Cryptology — The Science of Information Integrity*, G. J. Simmons Editor, pp. 43–64, IEEE Press, 92.

[153] J. Stern. *La Science du Secret*, Odile Jacob, Paris, 1998.

[154] J. Stern, S. Vaudenay. SVP: a Flexible Micropayment Scheme. In *Financial Cryptography*, Anguilla, British West Indies, Lecture Notes in Computer Science 1318, pp. 161–171, Springer-Verlag, 1997.

[155] J. Stern, S. Vaudenay. CS-Cipher. In *Fast Software Encryption'98*, Paris, France, Lecture Notes in Computer Science 1372, pp. 189–205, Springer-Verlag, 1998. (Voir annexe G.)

[156] D. R. Stinson. *Cryptographie, Théorie et Pratique*, (Traduction française de S. Vaudenay), International Thomson Publishing, 1996.

[157] R. M. Tanner. Explicit Concentrators from Generalized N-gons. In *SIAM Journal of Algebraic Discrete Methods*, vol. 5, pp. 287–293, 1984.

[158] A. Tardy-Corfdir, H. Gilbert. A Known Plaintext Attack of FEAL-4 and FEAL-6. In *Advances in Cryptology CRYPTO'91*, Santa Barbara, Californie, U.S.A., Lecture Notes in Computer Science 576, pp. 172–181, Springer-Verlag, 1992.

[159] T. Theobald. How to Break Shamir's Asymmetric Basis. In *Advances in Cryptology CRYPTO'95*, Santa Barbara, Californie, U.S.A., Lecture Notes in Computer Science 963, pp. 136–147, Springer-Verlag, 1995.

[160] S. Vaudenay. One-time identification with low memory. In *Proc. EUROCODE'92*, Udine, Italie, CISM Courses and lectures 339, pp. 217–228, Springer-Verlag, 1993.

[161] S. Vaudenay. FFT-Hash-II is not yet Collision-Free. In *Advances in Cryptology CRYPTO'92*, Santa Barbara, Californie, U.S.A., Lecture Notes in Computer Science 740, pp. 587–593, Springer-Verlag, 1993. (Voir annexe A.)

[162] S. Vaudenay. On the Need for Multipermutations: Cryptanalysis of MD4 and SAFER. In *Fast Software Encryption'94*, Leuven, Belgique, Lecture Notes in Computer Science 1008, pp. 286–297, Springer-Verlag, 1995. (Voir annexe D.)

[163] S. Vaudenay. *La Sécurité des Primitives Cryptographiques*, Thèse de Doctorat de l'Université de Paris 7, Technical Report LIENS-95-10 of the Laboratoire d'Informatique de l'Ecole Normale Supérieure, 1995.

[164] S. Vaudenay. On the weak Keys of Blowfish. In *Fast Software Encryption'96*, Cambridge, Royaume Uni, Lecture Notes in Computer Science 1039, pp. 27–32, Springer-Verlag, 1996. (Voir annexe B.)

[165] S. Vaudenay. An Experiment on DES — Statistical Cryptanalysis. In *3rd ACM Conference on Computer and Communications Security*, New Delhi, India, pp. 139–147, ACM Press, 1996. (Voir annexe C.)

[166] S. Vaudenay. Hidden Collisions on DSS. In *Advances in Cryptology CRYPTO'96*, Santa Barbara, Californie, U.S.A., Lecture Notes in Computer Science 1109, pp. 83–88, Springer-Verlag, 1996. (Voir annexe K.)

[167] S. Vaudenay. On the Security of Lenstra's DSA Variant. Présenté à la *Rump Session* de la conférence Asiacrypt'96.
`ftp://ftp.ens.fr/pub/dmi/users/vaudenay/lenstra.ps`.

[168] S. Vaudenay. Procédé de décorrélation de données. Brevet en France num. 96 13411 demandé le 04/11/1996.

[169] S. Vaudenay. Casser des Algorithmes. *Pour la Science*, num. 220, p. 54, 1996.

[170] S. Vaudenay. Towards Provable Security for Feistel Ciphers. In *SAC'96, Third Annual Workshop on Selected Areas in Cryptography*, Workshop Record, pp. 92–94, Queen's University, Kingston, Ontario, Canada, 1996.

[171] S. Vaudenay. A cheap Paradigm for Block Cipher Security Strengthening. Technical Report LIENS-97-3, 1997.

[172] S. Vaudenay. Un Réseau Quantique. *Pour la Science*, num. 234, p. 30, 1997.

[173] S. Vaudenay. Provable Security for Block Ciphers by Decorrelation. In *STACS 98*, Paris, France, Lecture Notes in Computer Science 1373, pp. 249–275, Springer-Verlag, 1998.

[174] S. Vaudenay. Provable Security for Block Ciphers by Decorrelation. (Journal Version.) Submitted. (Voir annexe F.)

[175] S. Vaudenay (Ed.). *Fast Software Encryption*, Proceedings, Lecture Notes in Computer Science vol. 1372, Springer-Verlag, 1998.

[176] S. Vaudenay. Feistel Ciphers with L_2-Decorrelation. To appear in SAC'98, LNCS.

[177] S. Vaudenay. The Decorrelation Technique Home-Page.
URL:`http://www.dmi.ens.fr/~vaudenay/decorrelation.html`

[178] S. Vaudenay. Cryptanalysis of the Chor-Rivest Cryptosystem. In *Advances in Cryptology CRYPTO'98*, Santa Barbara, Californie, U.S.A., Lecture Notes in Computer Science 1462, pp. 243–256, Springer-Verlag, 1998. (Voir annexe I.)

[179] S. Vaudenay. Cryptanalysis of the Chor-Rivest Cryptosystem. (Soumission — Version journal de [178].) (Voir annexe I.)

[180] S. Vaudenay, J. Stern, D. Sabban. Nouveau procédé de cryptographie numérique par boîte de mélange. Brevet en France demandé le 10/10/1997.

[181] G. S. Vernam. Cipher Printing Telegraph Systems for Secret Wire and Radio Telegraphic Communications. *Journal of the American Institute of Electrical Engineers*, vol. 45, pp. 109–115, 1926.

[182] R. Verser. Strong Cryptography Makes the World a Safer Place.
`http://www.frii.com/~rcv/deschall.htm`

[183] M. N. Wegman, J. L. Carter. New Hash Functions and their Use in Authentication and Set Equality. *Journal of Computer and System Sciences*, vol. 22, pp. 265–279, 1981.

Notations

$\mathbf{Z}_2$ alphabet binaire $\{0,1\}$. Par extension, groupe muni de la loi *ou* exclusif notée $\oplus$ et définie par $0 \oplus 0 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$, et $1 \oplus 1 = 0$. Par extension, corps muni de la loi $\oplus$ et de la multiplication usuelle. On a $a \oplus b = a + b \bmod 2$.

$a \oplus b$ *ou* exclusif bit-à-bit. Lorsque a et b sont des bits, c'est la loi du groupe $\mathbf{Z}_2$. Lorsque a et b sont des chaînes de bits de même longueur n, c'est la loi du groupe produit $\mathbf{Z}_2^n$ (voir p. 23).

$a \cdot b$ produit scalaire. Pour deux chaînes de bits de même longueur $a = a_1 \dots a_\ell$ et $b = b_1 \dots b_\ell$, c'est $(a_1 b_1 + \dots + a_\ell b_\ell) \bmod 2$. C'est le produit scalaire dans l'espace vectoriel $\mathrm{GF}(2)^\ell$ (voir Eq. (2.7) p. 47).

$a \bmod b$ opération modulo : reste dans la division euclidienne de a par b.

$a \equiv b \pmod{n}$ congruence modulo n : $a - b$ est un multiple de n.

$\lfloor x \rfloor$ plus grand entier inférieur ou égal à x.

$\mathbf{Z}_n$ ensemble $\{0, \dots, n-1\}$. Par extension, groupe des résidus modulo n muni de l'addition usuelle modulo n : dans $\mathbf{Z}_n$, la somme de a et de b est l'entier $a + b \bmod n$. Par extension, anneau des résidus modulo n muni de l'addition et de la multiplication usuelles modulo n. Lorsque n est un nombre premier, l'anneau $\mathbf{Z}_n$ est un corps et l'opération $1/a \bmod n$ est définie pour $a \not\equiv 0 \pmod{n}$.

$\mathcal{A} \times \mathcal{B}$ produit cartésien des ensembles $\mathcal{A}$ et $\mathcal{B}$: ensemble des couples (a, b) où $a \in \mathcal{A}$ et $b \in \mathcal{B}$. Par extension (lorsque $\mathcal{A}$ et $\mathcal{B}$ sont des groupes), groupe produit défini par la loi

$$(a, b) + (a', b') = (a + a', b + b').$$

$f : \mathcal{A} \to \mathcal{B}$ fonction f dont l'ensemble de départ est $\mathcal{A}$ et l'ensemble d'arrivée est $\mathcal{B}$.

$(x_i)_{i\in I}$ famille d'éléments x_i indexée par un indice i dans I. Cette notion est semblable à celle de fonction $i \mapsto x_i$ sur I.

$x \mapsto f(x)$ fonction qui à x associe $f(x)$ (en fait, f donc). Par exemple, $x \mapsto x+1$ est la fonction qui calcule le successeur de x dans le système de numération usuel.

$M_{i,j}$ pour une matrice M, terme de la ligne indexée par i et de la colonne indexée par j.

${}^tM$ transposée d'un opérateur linéaire M. De manière matricielle, si M est de type $m \times n$, alors ${}^tM$ est une matrice de type $n \times m$ définie par $({}^tM)_{i,j} = M_{j,i}$.

$E(X)$ espérance d'une variable aléatoire X.

$V(X)$ variance d'une variable aléatoire X.

$\Pr(E)$ probabilité d'un événement E.

$\Phi(x)$ fonction de distribution normale centrée. On a

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}} dt$$

(voir p. 55).

$\#A$ cardinal d'un ensemble A.

$x \leftarrow a$ opération d'affectation de la valeur a au registre x. Par exemple, $x \leftarrow x+1$ est l'opération qui incrémente le registre x.

$\Psi(g_1, \ldots, g_r)$ schéma de Feistel, voir Déf. 1.6 p. 23.

$\texttt{01b3c7}_x$ chaîne de bits hexadécimale. Lorsque la chaîne a une longueur multiple de quatre, il est pratique de la représenter en numération hexadécimale. Ainsi $\texttt{01b3c7}_x$ représente-t-il la chaîne

$$0000\,0001\,1011\,0011\,1100\,0111.$$

Lorsque la chaîne admet une autre longueur, on la complète à gauche par un, deux ou trois bits nuls pour pouvoir la représenter en numération hexadécimale en précisant explicitement sa longueur.

$a|b$ concaténation de deux chaînes. Si $a = a_1 \ldots a_m$ et $b = b_1 \ldots b_n$, alors $a|b$ est la chaîne $a_1 \ldots a_m b_1 \ldots b_n$.

$G = (V, E)$ graphe orienté ou non, voir Déf. 1.1 p. 21 pour les graphes orientés, et Déf. 2.17 p. 56 pour les graphes non orientés.

$G = (V, E, \mathrm{Dom})$ réseau de calcul, voir Déf. 1.2 p. 21.

$G = (V, E, \mathrm{Dom}, f)$ réseau de calcul interprété, voir Déf. 1.4 p. 21.

$G = (V, E, \mathrm{Dom}, \mathrm{Sol})$ graphe d'équations, voir Déf. 2.18 p. 56.

$G = (V, E, \mathrm{Dom}, \mathrm{Sol}, f)$ graphe d'équations interprété, voir Déf. 2.19 p. 57.

In_u ensemble des arêtes d'extrémité u dans un graphe orienté (voir Déf. 1.1 p. 21).

Out_u ensemble des arêtes d'origine u dans un graphe orienté (voir Déf. 1.1 p. 21).

In_G ensemble des arêtes entrantes d'un réseau de calcul G (voir Déf. 1.2 p. 21).

Out_G ensemble des arêtes sortantes d'un réseau de calcul G (voir Déf. 1.2 p. 21).

Dom_a domaine de définition d'une variable définie par une arête a d'un graphe (voir Déf. 1.2 p. 21).

$\mathrm{Dom}(A)$ ensemble des évaluations partielles définies sur A, voir Déf. 1.3 p. 21.

$t(a)$ valeur d'une arête a pour une évaluation partielle t, voir Déf. 1.3 p. 21.

$t|_A$ restriction d'une évaluation partielle à A, voir Déf. 1.3 p. 21.

f_u fonction associée au sommet u pour une interprétation f, voir Déf. 1.4 p. 21.

$\mathrm{Val}^G(x)$ évaluation d'un réseau de calcul G pour une entrée x, voir Déf. 1.5 p. 22.

$G(x)$ sortie de la fonction associée à un réseau de calcul G pour une entrée x, voir Déf. 1.5 p. 22.

Adj_u ensemble des arêtes adjacentes à un sommet u dans un graphe, voir Déf. 2.17 p. 56.

$\mathrm{Adj}(U)$ ensemble des arêtes adjacentes à au moins un sommet de l'ensemble U dans un graphe, voir preuve du Lem. 2.27 p. 61.

$E|_U$ ensemble des arêtes adjacentes à deux sommets de U, voir Lemme 2.27 p. 61.

Sol_u nombre de solutions d'une équation locale correspondant à un sommet u dans un graphe d'équations, voir Déf. 2.18 p. 56.

$\mathrm{Sol}(U)$ nombre de solutions d'un sous-ensemble d'équations locales correspondant à un ensemble U de sommets dans un graphe d'équations, voir Déf. 2.28 p. 62.

$X \bowtie Y$ jointure de deux ensembles de solutions partiels, voir Déf. 2.20 p. 57.

$\gamma_p(X)$ sélection aléatoire des éléments d'un ensemble X avec probabilité p, voir Déf. 2.21 p. 58.

σ_u solutions locales d'une équation u, voir Déf. 2.21 p. 58.

$\mathrm{Val}_f(R)$ résultat d'un algorithme de résolution R pour une interprétation f d'un graphe d'équations, voir Déf. 2.21 p. 58.

$V(R)$ ensemble de sommets représentés dans un algorithme de résolution R, voir Lemme 2.27 p. 61.

$\mathrm{Comp}_I(R)$ complexité d'un algorithme de résolution, voir Déf. 2.22 p. 59.

$\mathrm{Comp}'(R)$ complexité théorique d'un algorithme de résolution, voir Déf. 2.30 p. 63.

Γ forme quadratique associée à un graphe d'équations, voir Déf. 2.28 p. 62.

$\Gamma(g)$ valeur d'une forme quadratique associée à un graphe d'équations sur une fonction g, voir Déf. 2.28 p. 62.

$\Gamma(U)$ valeur d'une forme quadratique associée à un graphe d'équations sur un ensemble de sommets U, voir Déf. 2.28 p. 62.

Ω caractéristique d'un réseau de calcul, voir Déf. 2.1 p. 38.

E_Ω événement qui correspond à une caractéristique différentielle, voir Eq. (2.2) p. 39.

$\mathrm{DP}^f(a, b)$ probabilité différentielle. On définit

$$\mathrm{DP}^f(a, b) = \Pr_X[f(X \oplus a) \oplus f(X) = b]$$

pour une fonction déterminée f, voir Déf. 2.2 p. 38.

$\mathrm{DP}^G(\Omega)$ probabilité théorique d'une caractéristique, voir Déf. 2.3 p. 38.

$\mathrm{LP}^f(a,b)$ biais de linéarité. On définit

$$\mathrm{LP}^f(a,b) = \left(2 \Pr_X[a \cdot X = b \cdot f(X)] - 1\right)$$

pour une fonction déterminée f, voir Déf. 2.7 p. 46.

$\mathrm{LP}^G(\Omega)$ coefficient LP d'une caractéristique, voir Déf. 2.8 p. 47.

$\mathrm{DP}^f_{\max}$ maximum de DP^f, voir Eq. (3.1) p. 75.

$\mathrm{LP}^f_{\max}$ maximum de LP^f, voir Eq. (3.2) p. 75.

EDP^f espérance de DP^f, voir Eq. (3.3) p. 80.

ELP^f espérance de LP^f, voir Eq. (3.3) p. 80.

$\mathrm{EDP}^f_{\max}$ maximum de EDP^f, voir Eq. (3.3) p. 81.

$\mathrm{ELP}^f_{\max}$ maximum de ELP^f, voir Eq. (3.4) p. 81.

$[F]^d$ matrice de décorrélation à l'ordre d d'une fonction aléatoire F, voir Déf. 3.13 p. 82.

$\mathrm{DecF}^d(F)$ biais de décorrélation d'ordre d de la fonction aléatoire F, voir Déf. 3.14 p. 82.

$\mathrm{DecP}^d(C)$ biais de décorrélation d'ordre d de la permutation aléatoire C, voir Déf. 3.14 p. 82.

$|||A|||_\infty$ norme de la matrice A associée à L_∞, voir Eq. (3.5) p. 83.

$||A||$ norme de la matrice A. Dans ce rapport, c'est la norme $|||A|||_\infty$.

Index

Curriculum Vitæ

Etat Civil

Serge Vaudenay, né le 5 avril 1968 à Saint-Maur des Fossés, marié sans enfant.

Formation

1989 Intégration à l'*Ecole Normale Supérieure*.

1989-92 *Magistère MMFAI* regroupant plusieurs diplômes avec mention très bien en mathématiques et en informatique, dont un DEA spécialisé en complexité et cryptographie.

1992 Réussite au concours de l'*Agrégation* de mathématiques (rang 26).

1991-95 Doctorat dirigé par Jacques Stern sur la sécurité des fonctions cryptographiques primitives. Diplôme obtenu le 26 mai 1995 avec les félicitations du jury.

Cursus Professionnel

1989-93 Elève de l'*Ecole Normale Supérieure*, fonctionnaire stagiaire.

1993-94 Service national actif (scientifique du contingent).

1994-95 Allocateur moniteur à l'*Ecole Normale Supérieure*.

1995- Chargé de recherche au CNRS, affecté au laboratoire d'informatique de l'*Ecole Normale Supérieure*, fonctionnaire (titularisé le 1er avril 1997).

Enseignement

1989-91 Colleur en mathématiques au *lycée Fénelon* et au *lycée Saint-Louis*. (Mathématiques supérieures puis mathématiques spéciales.)

1994 Cours d'informatique à l'*IMAC*. (Cryptographie.)

1994-96 Travaux pratiques d'informatique à l'*Ecole Polytechnique* (vacataire). (Introduction à la programmation.)

1994-98 Travaux dirigés d'informatique à l'*Ecole Normale Supérieure*. (Suivant les années : algorithmique et complexité, programmation système et architecture des ordinateurs.)

1998-99 Chef de Travaux Pratiques à l'*Ecole Polytechnique*. (Programmation en Java.)

Stages de Recherche

Eté 1989 Stage de maîtrise à l'*INRIA-Sophia-Antipolis*. Recherches dirigées par Jean-Daniel Boissonnat et Olivier Devillers : création d'un programme C pour le tracé de diagrammes de Voronoï par un algorithme incrémental.

Eté 1991 Stage de DEA dans l'entreprise *GEMPLUS Card International*. Recherches sur la mise au point de protocoles *zero-knowledge* pour des cartes à puce bas-coût. (Langage machine 6805.)

Eté 1992 Stage de recherche à l'*Université de Montréal*. Recherches avec Gilles Brassard et Claude Crépeau.

Eté 1995 Stage de recherche à l'*Université de Carleton*. Recherches avec Evangelos Kranakis et Mike Just.

Février 1996 Invité au *Computer Security, Cryptography and Coding Theory Programme* de l'Institut Isaac Newton.

Mars 1996 Obtention d'une bourse de la *Japanese Society for Promotion of Sciences* pour effectuer des recherches à l'*Université de Yokohama* avec Tsutomu Matsumoto.

Eté 1997 Invité aux laboratoires *AT&T Research* par Andrew Odlyzko.

Février 1999 Invité aux laboratoires de *NTT* par Tatsuaki Okamoto.

Valorisation de la Recherche

1995 Membre du comité de programme de Crypto'95.

1996 Membre du comité de programme de Eurocrypt'96.

Conférencier invité au colloque SAC'96.

Conférencier invité à l'"Ecole d'automne AMIcale".

1997 Organisateur du *Security Protocol Workshop'97* et membre du comité de programme.

1998 Conférencier invité au colloque STACS'98

Organisateur du colloque *Fast Sofware Encryption'98*, président du comité de programme et éditeur des actes.

Membre du comité de programme de Eurocrypt'98.

Membre du comité de programme de SAC'98.

1999 Membre du comité de programme de Fast Software Encryption'99.

Membre du comité de programme de Crypto'99.

Diffusion de la Recherche

1995 Traduction en français du livre de Douglas Stinson *Cryptographie, Théorie et Pratique*.

Administration

1996- Membre de la *Commission des Etudes* de l'Ecole Normale Supérieure.

Membre du *Conseil de Laboratoire* du LIENS, unité de recherche associée au CNRS.

1998- Membre de la *Commission des Spécialistes* de l'Ecole Normale Supérieure en section informatique.

Membre de la *Commission des Spécialistes* de l'Université de Paris-Sud en section informatique.

Liste de Publications

Thèses — Livres

[163] S. Vaudenay. *La Sécurité des Primitives Cryptographiques*, Thèse de Doctorat de l'Université de Paris 7. 1995.

[156] D. R. Stinson. *Cryptographie, Théorie et Pratique* (Traduction française de S. Vaudenay). 1996.

[175] S. Vaudenay (Ed.). *Fast Software Encryption*, Proceedings, Lecture Notes in Computer Science vol. 1372, Springer-Verlag, 1998.

Articles de Journaux

[40] D. Coppersmith, J. Stern, S. Vaudenay. The Security of the Birational Permutation Signature Schemes. *Journal of Cryptology*, 1997.

[147] C. P. Schnorr, S. Vaudenay. Black Box Cryptanalysis of Cryptographic Primitives. *Journal of Cryptology*, 1998.

[174] S. Vaudenay. Provable Security for Block Ciphers by Decorrelation. (Soumission — Version journal de [173].)

[179] S. Vaudenay. Cryptanalysis of the Chor-Rivest Cryptosystem. (Soumission — Version journal de [178].)

Actes de Colloques avec Comité de Lecture

Série Eurocrypt

[35] F. Chabaud, S. Vaudenay. Links between Differential and Linear Cryptanalysis. Eurocrypt 94.

[108] D. Naccache, D. M'Raïhi, D. Raphaeli, S. Vaudenay. Complexity Trade-offs with the Digital Signature Standard. Eurocrypt 94.

[146] C.-P. Schnorr, S. Vaudenay. Black Box Cryptanalysis of Hash Networks based on Multipermutations. Eurocrypt 94.

Série Crypto

[161] S. Vaudenay. FFT-Hash-II is not yet Collision-Free. Crypto 92.

[39] D. Coppersmith, J. Stern, S. Vaudenay. Attacks on the Birational Permutation Signature. Crypto 93.

[166] S. Vaudenay. Hidden Collisions on DSS. Crypto 96.

[178] S. Vaudenay. Cryptanalysis of the Chor-Rivest Cryptosystem. Crypto 98.

Série Fast Software Encryption

[145] C.-P. Schnorr, S. Vaudenay. Parallel FFT-Hashing. *Fast Software Encryption* 93.

[162] S. Vaudenay. On the Need for Multipermutations : Cryptanalysis of MD4 and SAFER. *Fast Software Encryption* 94.

[164] S. Vaudenay. On the Weak Keys of Blowfish. *Fast Software Encryption* 96.

[105] D. M'Raïhi, D. Naccache, J. Stern, S. Vaudenay. xmx - a Firmware-Oriented Block Cipher based on Modular Multiplications. *Fast Software Encryption* 97.

[155] J. Stern, S. Vaudenay. CS-Cipher. *Fast Software Encryption* 98.

Série Asiacrypt

[69] M. Just, S. Vaudenay. Authenticated Multi-Party Key Agreement. Asiacrypt 96.

[16] R. Anderson, S. Vaudenay. Minding your p's and q's. Asiacrypt 96.

Autre Séries LNCS

[17] R. Anderson, S. Vaudenay, B. Preneel, K. Nyberg. The Newton Channel. *Information Hiding* 96.

[127] G. Poupard, S. Vaudenay. Decorrelated Fast Cipher : an AES Candidate well Suited for Low Cost Smart Cards Applications. Cardis 98.

[154] J. Stern, S. Vaudenay. SVP : a Flexible Micropayment Scheme. *Financial Cryptography* 97.

[173] S. Vaudenay. Provable Security for Block Ciphers by Decorrelation. Stacs 98.

[176] S. Vaudenay. Feistel Ciphers with L_2-Decorrelation. A paraître dans SAC 98.

Série ACM Computer and Communication Security

[165] S. Vaudenay. An Experiment on DES — Statistical Cryptanalysis. ACM *Computer and Communications Security* 96.

Autres Séries

[160] S. Vaudenay. One-Time Identification with low Memory. Eurocode 92.

[55] H. Gilbert, M. Girault, P. Hoogvorst, F. Noilhan, T. Pornin, G. Poupard, J. Stern, S. Vaudenay. Decorrelated Fast Cipher : an AES Candidate. First Advanced Encryption Standard Candidate Conference, 1998.

Articles de Vulgarisation

[169] S. Vaudenay. Casser des Algorithmes. *Pour la Science*, 1996.

[172] S. Vaudenay. Un Réseau Quantique. *Pour la Science*, 1997.

[99] F. Mébarki. Secret de Deux, Secret de Dieu. *Le journal du CNRS*, 1997.

[10] Méthode de Chiffrement Fondée sur la Décorrélation. *100 Faits Marquants du Département des Sciences Pour l'Ingénieur*, CNRS, 1997.

Brevets

[106] D. M'Raïhi, S. Vaudenay. Procédé de Cryptographie de Messages Transmis par un Support d'Informations à un Système de Traitement. 1992.

[104] D. M'Raïhi, D. Naccache, J. Stern, S. Vaudenay. Procédé de Cryptographie à Clé Publique basé sur le Logarithme Discret. 1995.

[168] S. Vaudenay. Procédé de Décorrélation de Données. 1996.

[180] S. Vaudenay, J. Stern, D. Sabban. Nouveau Procédé de Cryptographie Numérique par Boîte de Mélange. 1997.

Autres

[167] S. Vaudenay. On the Security of Lenstra's DSA Variant. *Rump Session* de Asiacrypt 96.
`ftp://ftp.ens.fr/pub/dmi/users/vaudenay/lenstra.ps`.

[170] S. Vaudenay. Towards Provable Security for Feistel Ciphers. SAC 96.

[125] D. Pointcheval, S. Vaudenay. On Provable Security for Digital Signature Algorithms. Rapport LIENS 96-17, 1996.

[171] S. Vaudenay. A Cheap Paradigm for Block Cipher Security Strengthening. Rapport LIENS 97-3, 1997.

[56] H. Gilbert, M. Girault, P. Hoogvorst, F. Noilhan, T. Pornin, G. Poupard, J. Stern, S. Vaudenay. Decorrelated Fast Cipher : an AES Candidate. Soumission au processus Advanced Encryption Standard.

Table des Matières

Motivée par le commerce et l'industrie, la recherche publique dans le domaine du chiffrement symétrique s'est considérablement développée depuis vingt cinq ans si bien qu'il est maintenant possible d'en faire le bilan.

La recherche a tout d'abord progressé de manière empirique. De nombreux algorithmes de chiffrement fondés sur la notion de réseau de substitutions et de permutations ont été proposés, suivis d'attaques dédiées contre eux. Cela a permis de définir des stratégies générales : les méthodes d'attaques différentielles, linéaires et statistiques, et les méthodes génériques fondées sur la notion de boîte noire.

En modélisant ces attaques on a trouvé en retour des règles utiles dans la conception d'algorithmes sûrs : la notion combinatoire de multipermutation pour les fonctions élémentaires, le contrôle de la diffusion par des critères géométriques de réseau de calcul, l'étude algébrique de la non-linéarité, ... Enfin, on montre que la sécurité face à un grand nombre de classes d'attaques classiques est assurée grâce à la notion de décorrélation par une preuve formelle.

Ces principes sont à l'origine de deux algorithmes particuliers : la fonction CS-Cipher qui permet un chiffrement à haut débit et une sécurité heuristique, et le candidat DFC au processus de standardisation AES, prototype d'algorithme fondé sur la notion de décorrélation.